Onions in the Queue: An Integral Networking Perspective on Anonymous Communication Systems

DISSERTATION
zur Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von
Florian Tschorsch

Präsident der Humboldt-Universität zu Berlin
Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät
Prof. Dr. Elmar Kulke

1. Gutachter: Prof. Dr. Björn Scheuermann
2. Gutachter: Prof. Dr. Odej Kao
3. Gutachter: Prof. Dr. Martin Mauve

Tag der Einreichung: 8. März 2016
Tag der mündlichen Prüfung: 6. Juni 2016

Onions in the Queue: An Integral Networking Perspective on Anonymous Communication Systems

Florian Tschorsch

ABSTRACT

Performance is a pivot point in the design of anonymity overlays. Due to their growing popularity, they are faced with increasing load, which makes design problems imminent. The special requirements and complex architecture of anonymity overlays renders the topic a challenging but likewise inspiring object of research.

In this work, we discuss the design of low-latency anonymous communication systems in general and the Tor network as the de-facto standard in particular. We develop solutions to a number of research questions, all collectively following the aim of enhancing and securing such networks. By doing this we create a fundamental technical understanding of networking aspects in anonymity overlays and tackle the most prevalent performance issue experienced today: network congestion.

To this end, we systematically explore the design space of data transport in anonymity overlays and reveal serious performance issues. This exploration provides insights in how (not) to design a transport protocol for anonymity overlays. In order to support future design decisions, we additionally present a methodology to measure networks in a privacy-preserving manner.

The fundamental results of this thesis include the discovery of a destructive denial of service attack and the associated design flaw of performing hop-by-hop reliability and end-to-end flow control. Moreover, we emphasize the central role of fairness. In particular, we show that gross unfairness between circuits may arise and lead to poor performance. While these kind of issues are difficult to fix, we provide respective security measures and a fully distributed scheduling algorithm that implicitly achieves global fairness.

These issues clearly demonstrate the inadequacy of currently employed congestion control in anonymity overlays. In particular, we identify a feedback "gap" between incoming and outgoing connections as the primary cause of performance issues. Aware of the requirements and the problems of anonymity overlays, we develop a tailored transport protocol. It combines congestion control with a backpressure-based flow control mechanism. The resulting overlay is able to react locally and thus rapidly to varying network conditions. It yields superior performance and a resilient overlay network.

With our work, we contribute an integral perspective on networking aspects of anonymity overlays and tackle the root cause of performance issues.

ZUSAMMENFASSUNG

Performanz ist ein zentraler Bestandteil des Designs von Anonymisierungsdiensten. Ihre zunehmende Popularität führt jedoch zu einer hohen Netzwerklast, die unzulängliche Entwurfsentscheidungen imminent macht. Die Anforderungen und die vielschichtige Architektur von Anonymisierungsdiensten machen die Thematik zu einem anspruchsvollen und zugleich inspirierenden Forschungsgegenstand.

Die vorliegende Arbeit diskutiert das Design von sogenannten Niedriglatenz-Anonymisierungsdiensten im Allgemeinen und dem Tor-Netzwerk als relevantesten Vertreter im Speziellen. Es werden Lösungen für eine Reihe von Forschungsfragen entwickelt, die allesamt das Ziel verfolgen, diese Overlay-Netzwerke zu verbessern und sicherer zu gestalten. Es entsteht ein fundamentales Verständnis zu Netzwerkaspekten in Anonymisierungs-Overlays, das die Netzwerklast, als vorherrschende Ursache für die schwache Performanz, thematisiert.

Dazu werden die Freiheitsgrade im Design des Datentransports systematisch untersucht und folgenreiche Schwachpunkte aufgezeigt. Diese Betrachtung liefert Erkenntnisse darüber, wie eine Transportschicht für Anonymisierungs-Overlays (nicht) konstruiert werden sollte. Um zukünftige Entwicklungen zu unterstützen wird zusätzlich eine Methodik zur Erhebung von Netzwerkstatistiken präsentiert, die die Privatsphäre der Nutzer weiterhin schützt.

Zu den fundamentalen Ergebnissen dieser Arbeit zählen die Aufdeckung eines destruktiven Denial-of-Service-Angriffs und des zugehörigen Designfehlers, der darin besteht, dass Zuverlässigkeit Hop-zu-Hop und Flusskontrolle Ende-zu-Ende implementiert werden. Außerdem wird die zentrale Rolle von Fairness erkannt. Insbesondere wird gezeigt, dass massive Unfairness zwischen Circuits entstehen und zu schlechter Performanz führen kann. Während diese Schwachstellen nicht ohne Weiteres zu beheben sind, werden entsprechende Sicherheitsmechanismen und ein verteilter Scheduler, der implizit globale Fairness erreicht, vorgestellt.

Eine „Lücke“ zwischen den Feedbackmechanismen von ein- und ausgehenden Verbindungen wird schließlich als Hauptgrund für die unzulängliche Lastkontrolle von Anonymisierungs-Overlays identifiziert. Unter Berücksichtigung der erworbenen Erkenntnisse wird ein maßgeschneidertes Transportprotokoll entwickelt, das Überlastkontrolle mit einer rückdruck-basierten Flusskontrolle kombiniert. Das daraus resultierende Overlay ist in der Lage lokal auf wechselnde Netzwerkbedingungen zu reagieren und liefert eine verbesserte Performanz und ein robustes Netzwerk.

Der Beitrag der vorliegenden Arbeit ist eine integrale Netzwerkperspektive auf Anonymisierungs-Overlays, die den Kern der Performanz-Probleme adressieren.

PREFACE AND ACKNOWLEDGMENTS

This thesis concludes, quite literally, a research expedition. It would not have been possible, though, without the support of my colleagues, friends, and family. While the attempt to mention everyone by name would be bound to fail, I would like to reflect on the past years.

In December 2010, I joined Björn Scheuermann's research group at the Heinrich Heine University Düsseldorf, which marks the beginning of my journey. Since then, we moved three times with stopovers at University of Würzburg and University of Bonn, before we finally settled at Humboldt University of Berlin in October 2012. Along the route, people left and joined our research group (a. k. a. the research "caravan"). In fact, I am the only one who outlasted the whole journey—and that is mainly because of Björn. He gave me the opportunity to pursue my research on an exciting topic and taught me what it takes to develop and sell an idea. Björn, thank you for being an honest, dedicated and constructive advisor.

I also want to thank my other two reviewers, Odej Kao and Martin Mauve, as well as my graduation committee members, Wolfgang Reisig, Eckhard Grass, and Sven Hager. Particularly, Odej and Martin challenged me with profound questions. Odej and Martin, thank you for taking the time to work through my thesis and writing such elaborate reviews.

During my journey, I worked with many inspiring colleagues. The exchange with my colleagues nurtured my creativity and improved the quality of my work. Till and Christian, thank you for an unusual summer in Würzburg with an excursion into the field of telematics and control theory. Evgeni, thank you for long conversations on the train while commuting between Düsseldorf and Bonn. Sven, thank you for educating me in the fine art of compilers and software development. Stefan, thank you for discussing my thesis and for questioning every sentence of my thesis' introduction. Your sharp comments really helped me focus. Daniel, thank you for the fun times and all the laughter. Sharing the office with you for many years now is priceless—except for the thugs coming for the Bitcoin miner.

In the past years, the Tor project's public image changed from a nefarious tool to an influential network. I am proud to see that my humble contributions had an impact on Tor. I thank Rodger Dingledine for introducing me to the Tor community and building bridges to fellow researchers and developers. In particular to Karsten Loesing, Aaron Johnson, and Rob Jansen, who I want to acknowledge by name. It was a pleasure to collaborate with you.

I have (co-)supervised as many as 28 students' theses and thus had the opportunity to discuss approaches with many aspiring and talented students. I also worked with many research assistants who helped implementing ideas. Representatively for many others, Tobias Schall deserves my gratitude for his commitment. Many lines of code used for this thesis' evaluations carry his signature.

Finally, I thank my family. It is little, and different, but still the best family one can wish for. I received endless support in all situations and matters of life. My mother, Monika Tschorsch, is a strong woman who I admire for her positive attitude towards life. Mama, ich danke dir, Oma und Papa für eure herzliche Fürsorge und eure Unterstützung. Ihr habt mir Werte wie Ehrgeiz, Selbstvertrauen und Neugier vermittelt und damit eine hervorragende Ausgangssituation für mich geschaffen. Ich bin mindestens genauso stolz auf euch, wie ihr auf mich. My deepest gratitude, though, deserves my wife, Rebecca Tschorsch. She endured not only the journey but also accepted a stressed me, covered my back, and provided a happy place. Becci, thank you for your tolerance and understanding, for your unconditional support and loyalty, and for your love. Well, thank you for being you.

I think I am ready for another adventure.

Florian
Berlin, June 2016

Cast

Advisor & Reviewer 1	Björn Scheuermann
Reviewer 2	Odej Kao
Reviewer 3	Martin Mauve
Committee chairman	Wolfang Reisig
Committee member	Eckhard Grass
	Sven Hager
Co-author	Rob G. Jansen
	Aaron Johnson
	Daniel Marks
Colleague	Stefan Dietzel
	Daniel Cagara
Research assistant	Tobias Schall

Crew

Ohana	Rebecca Tschorsch
	Monika Tschorsch
	Susanne Weber
	Heinz Jansen

In loving memory of
Maria Wessel and Kurt Tschorsch

Colleagues

Samuel Brack, Daniel Cagara, Stefan Dietzel, Roger Dingledine, Tim Dittler, Holger Doebler, Till Elsner, Wladislaw Gusew, Evgeni Golov, Sven Hager, Sebastian Henningsen, Christian Herrmann, Rob G. Jansen, Aaron Johnson, Karsten Loesing, Roman Naumann, Jedrzej Rybicki, Siegmar Sommer, Michael Stini, Steffen Tschirpke, Frank Winkler

Research Assistants

Tobias Amft, Christoph Bode, Markus Krug, Johannes Michael Mühr, Matthias Radig, Tobias Schall

Students

Christoph Bode, Daniel Cagara, Tim Dittler, Mirko Dressler, Pascal Grün, Arne Hoffmann, Emilio Janzen, Christian Klopp, Jann-Frederik Laß, Tim Repke, Elias Rohrer, Manuel Rüger, Marie Schaeffer, Tobias Schall, Phillipp Schoppmann, Philipp Sessler, Hagen Sparka, Fabio Tacke, Moritz Wedel

Universities

Funding

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

The Internet often conveys the impression of providing an anonymous communication channel, but in fact the underlying infrastructure was not designed with anonymity in mind: IP addresses serve as identifiers and are generally accessible by the recipient of a message as well as anybody with access to the communication channel along the route. In addition, there is less reason to believe that we can trust the network and its systems to behave as desired. The loss in trust becomes apparent due to an array of security and privacy breaches, such as denial-of-service and man-in-the-middle attacks, packet injection, censorship, and eavesdropping. Therefore, the basic assumption should be that all Internet communication is monitored and analyzed.

The attempt to enable anonymous Internet communication, that is, blending and obfuscating communication, leads away from simple direct communication between end systems. To this end, additional end systems, which act as intermediaries and augment application protocols are introduced [41]. As a result, overlay networks, which use the Internet as an infrastructure, evolve [54]. The concept of *anonymity overlays* is, therefore, not only to protect the content of messages, but also to obfuscate the exchange of messages itself. By routing data through a series of proxy servers, also termed *relays*, anonymity overlays distribute trust to independently-controlled relays and make sure no single relay learns both source *and* destination of a message. Eventually, anonymity is achieved if communication can be hidden in a set of other indistinguishable communications, which is commonly summarized by the notion of an anonymity set [162].

The design of anonymity overlays has its disadvantages, though. Because traffic needs to be relayed through multiple end systems, performance becomes an issue. Poor performance not only hinders wider adoption of anonymous communication [119], but also has a huge impact on the strength of anonymity: when more users join the network, everyone profits from an increased anonymity set. This matter of fact is expressed in the phrase "anonymity loves company" [70]. If users are discouraged by poor performance from using an anonymity service, though, the anonymity set of everyone shrinks and thus makes the degree of anonymity weaker. Therefore, good performance is a driving factor in the design of anonymity networks.

1.2 CHALLENGES AND OBJECTIVES

The Tor network [72] has become the prime example for anonymity overlays. Even in the presence of ubiquitous surveillance, it is currently the most effective service to preserve online anonymity [32]. Implementing what has become the standard architecture for low-latency anonymous communication systems [93], it routes application layer data along a cryptographically secured virtual *circuit* through an overlay network. Because of Tor's high relevance, it serves as our main object of research.

Unfortunately, Tor's current overlay design suffers from severe performance issues. More often than not, previous work on improving performance issues focused on isolated symptoms: for instance, data that dwells too long in socket buffers [25, 108, 166], a too rigid sliding window mechanism [24, 199], or service degradation caused by different traffic patterns [22, 110, 189].

We note that all of the named problems boil down to unsuitably chosen or unfavorably parameterized algorithms for congestion control, scheduling, and queue management. While these reasons have been pointed out before [73, 166], a consistent and superior overall design—beyond merely treating the symptoms—is still missing. Therefore, this thesis takes an integral perspective on performance aspects of transport in anonymity overlays. In particular, by conducting a comprehensive investigation of the transport design, we aim to unveil and solve the primary causes of performance problems.

Solving those problems is a challenging task, because the technical properties and requirements of anonymity overlays are specific. First, anonymity demands that relays used along one circuit should be located in different legislations and different autonomous systems. This condition typically implies long latencies, significantly longer than typical latencies for direct, non-anonymized communication. Consequently, response times are inherently slow. Second, anonymity and security requirements demand careful consideration of protocol designs. For example, control messages must not reveal user identities, neither directly nor indirectly. Third, relays are end systems, i. e., the access link is shared by all connections to and from one node. Therefore, overlay connections are, very much unlike the links of a router in a physical network, not independent. In the design of anonymity overlays, it is therefore important to consider not only the data transfer within the overlay, but also potential interference of overlay connections.

In addition to these challenges, solutions to the performance issues face another challenge: improved performance likely attracts more users, which strengthen anonymity. However, more users also imply relatively less resources for each and every one and hence inferior performance than with fewer users. It should also be noted that a large

fraction of all users' traffic is bulk traffic mostly produced by peer-to-peer file sharing applications [13, 143]. Therefore, any protocol design must be able to handle high load and yield a resilient network.

Our research aims to improve the performance of anonymity overlays in general and Tor as the de-facto standard in particular. While exploring the intrinsically complex design space, we raise attention to networking aspects. In particular, we are interested in the interrelations between underlay transport, packet scheduling, and overlay transport, with respect to throughput, delay, and fairness. Security and privacy aspects must be taken into account all the time. However, only from a rigorous networking perspective, we will be able to design a tailored transport protocol that tackles those performance issues.

Therefore, we assert the following thesis statement: *In order to tackle the primary causes of the prevalent performance issues in low-latency anonymous communication systems, an integral perspective on networking aspects is inevitable.*

1.3 OUTLINE AND CONTRIBUTIONS

We start by reviewing the state of the art in anonymous Internet communication and discussing related work in Chapter 2. Subsequently, we explore the design space of anonymity overlays and analyze design tradeoffs from various angles with an in-depth discussion on data transport. In particular, we emphasize that the respective protocol mechanisms are part of the application and thus of the application layer protocol typically running on *end systems*. The key contribution in Chapter 3 is an analysis using standard performance metrics, such as throughput, delay, and loss rate, which reveal non-obvious interactions. It shows that anonymity overlays cannot deliver satisfactory performance with any so far considered combination of overlay and underlay protocols and provides valuable insights in how (not) to design a transport protocol for anonymity overlays.

Anonymity overlays are inherently good at hiding information, what makes measuring the network a challenging task [96, 132, 134]. At the same time, though, the network's scale is of particular interest and provides relevant data to assess its effectiveness. This leads to the elementary question of how to determine the number of distinct users who contacted a service, while maintaining privacy. In Chapter 4, we present a methodology of network measurements in a distributed and privacy-preserving manner. We use probabilistic data structures as a basis and exploit their probabilistic nature to achieve privacy. In our analysis, we use the relative knowledge gain of an attacker as a privacy metric. This novel, information-theoretic viewpoint makes design tradeoffs more evident and more transparent. Based on arguments obtained with this methodology, we will demonstrate that,

even in the worst case, the knowledge gain of an attacker is limited when our distributed algorithm is employed.

Subsequently in Chapter 5, we demonstrate that Tor's currently employed congestion and flow control protocol exhibits severe deficiencies, which enable a destructive denial of service (DoS) attack against the Tor network. This novel attack, which we call the *Sniper Attack*, may be used to selectively disable arbitrary Tor relays with devastating implications. It forces a relay to buffer an excessive amount of data in application queues until it approaches the out-of-memory state and eventually gets terminated. Besides exposing (and fixing) a destructive attack vector in the most relevant anonymity overlay today, the main contribution includes the identification of a fundamental design issue. That is, performing hop-by-hop reliability and end-to-end flow control is an unfavorable design decision, which inevitably implies vulnerabilities and other disadvantages.

Chapter 6 is specifically dedicated to the aspect of fairness between circuits. This topic has almost entirely been neglected so far, despite being of particular importance: we show that, in the current Tor design, gross unfairness between circuits may arise and lead to poor performance. In order to overcome the unfairness, we introduce and discuss a resource allocation model based on *max-min fairness*. Furthermore, we show that max-min fair scheduling fits very well into an anonymity overlay and that the approach can achieve good resource utilization in conjunction with local and global fairness. This leads us to a redesign of Tor's scheduling algorithm. To this end, we contribute a fully distributed solution that tackles the local unfairness in Tor and implicitly achieves max-min fairness in the network without a need for explicit control messages or rate calculations. This is the foundation to yield a resilient overlay, which is able to handle the load.

The so far considered aspects reveal a number of fundamental issues in the design of Tor and lead to another imminent problem: Tor's flow and congestion control mechanisms. The fixed-size coarse-grained end-to-end sliding window, quite obviously, lacks adaptivity. As a result, excessive amounts of data can (and often will [64, 108, 166]) pile up in the per-circuit queues and in the socket buffers of a relay, i. e., in the "gap" between the incoming and outgoing TCP connections. We identify this gap as the primary cause of performance issues in Tor.

In Chapter 7 we take the acquired insights and develop techniques from the previous chapters to design a tailored solution, which we call *BackTap: Backpressure-based Transport Protocol*. Through per-hop flow control on circuit granularity, we allow the upstream node to control its sending behavior according to variations in the queue size of the respective downstream node. The result is backpressure that propagates along the circuit towards the source if a bottleneck is en-

countered. Since we implement (hop-by-hop) reliability in the application layer, instead of using TCP's reliability mechanisms between relays, relays in our architecture have a choice whether to accept or to drop data on a per-circuit basis. This avoids reliability-related security flaws as found in the Sniper Attack. We also do not use a fixed window size, neither end to end nor per hop. Instead, we adjust the per-hop window size using an appropriately adapted delay-based congestion control algorithm. What is often seen as a weakness of delay-based congestion control [15, 46]—being less aggressive than its loss-based counterparts—becomes a strength in our approach, because the aggressiveness of aggregate Tor traffic can otherwise be a significant problem. In packet level simulations we confirmed the expected improvement of BackTap.

In Chapter 8 we conclude this thesis.

1.4 SUMMARY OF THE MAIN CONTRIBUTIONS

The main scientific contributions of this thesis can be summarized as follows:

1. We unveil strong interrelations of overlay and underlay transport, most notably the influence of concurrent overlay connections on throughput, loss rate, and delay, which can lead to undesired performance penalties. In this context, we provide a network-oriented assessment of the design space of anonymity overlays and show that anonymity overlays are prone to such effects.

2. We present a privacy-preserving methodology to network measurement with the highest degree of privacy in mind, i. e., user statistics in anonymity overlays. With the analysis of our probabilistic approach, we provide a novel viewpoint on the degree of anonymity, which makes design tradeoffs more evident and transparent.

3. With the Sniper Attack we reveal a devastating threat to the Tor network and at the same time draw attention to a fundamental design flaw, i. e., separating the mechanisms for reliability and flow control in a transport protocol inevitably implies vulnerabilities. Even though we successfully implement and deploy a practical defense mechanism, it emphasizes Tor's unfavorable design decisions regarding data transport.

4. We develop a resource allocation scheme, particularly suitable for the specifics of anonymity overlays, i. e., our approach operates completely locally to enforce fairness between circuits on a global scale. Furthermore, we identify the notion of fairness as a key point to yield a resilient and effective overlay network.

5. We identify a feedback "gap" in Tor's protocol design as the primary cause for performance issues. Taking the acquired knowledge, we develop a backpressure-based transport protocol, namely BackTap, which clearly yields improved performance. BackTap is the first tailored transport protocol for the specifics of anonymity overlays, which aims for a holistic approach beyond treating isolated symptoms.

Despite being of scientific novelty, our contributions have practical relevance. They range from experimental proposals, over ready-to-implement solutions, to deployed algorithms.

2 STATE OF THE ONION

2.1 OVERVIEW

The special requirements and complex architecture renders anonymity overlays a challenging but likewise inspiring object of research. In this chapter, we introduce the state of the art in anonymous Internet communication. This includes a brief discussion on the relevance of traffic analysis on the Internet and a description of the onion routing protocol as a remedy to traffic analysis.

Based on the understanding of onion routing, we outline the main design challenge of anonymity overlays, i. e., the balance between anonymity and performance. In this context, we review the scientific discussion on the role of performance in anonymous communication and discuss related work relevant to all parts of this thesis. Related work referring to a special domain or aspect of our work will be discussed later in the respective chapter.

As a result of this chapter, a call for action regarding the shortcomings of the transport layer in anonymity overlays becomes apparent.

2.2 TRAFFIC ANALYSIS ON THE INTERNET

The Internet is a distributed network consisting of many voluntarily interconnected autonomous systems. Its design evolved from a number of research projects in the 1960s [126], and even though technology is changing quickly, the influences of those early days are still clearly visible [55]: the idea of internetworking accelerated the initial development. Most notably, the Internet Protocol (IP) became the building block that provides a "best effort" service but makes no guarantees about quality and reliability. The design decisions shaped the metaphor of an hourglass [20], where IP represents the waist and both below and above unfold a variety of protocols. Thus, IP is virtually ubiquitous on the Internet.

One of the reasons Internet communication is not considered anonymous is the ability to perform *traffic analysis*. Traffic analysis is the process of inferring information from communication even when the messages are encrypted and cannot be decrypted. In communications intelligence (COMINT) it refers to the concept of analyzing the technical metadata (also known as non-content). On the Internet, packet headers are usually not encrypted and serve as the foundation for traffic analysis. For example, source and destination IP addresses serve as identifier and can be used to trivially infer information such

as the location, contacts, duration, and frequency of communication. Furthermore, traffic analysis can be used to determine the operating system of the entities in a communication [185] or the password length of an SSH session [186].

In contrast to cryptanalysis, i. e., systematically recovering the contents of encrypted messages, traffic analysis is much "cheaper" to perform. It enables automated collection and bulk processing of data. Given the enormous amount of Internet communication and the value of information that can be obtained from traffic analysis, it does not come as a surprise that the "economics of surveillance" are relevant and applicable to the Internet as well [60]—that is, traffic analysis is employed in a large-scale. As a consequence, the basic assumption should be that all Internet communication is monitored and analyzed.

2.3 ONION ROUTING

As early as in 1981, David Chaum defined the traffic analysis problem as the problem of keeping who converses with whom and when confidential [51]. At the same time, Chaum introduced the cornerstone of anonymous Internet communication, i. e., mix networks, as a solution to the problem. He assumes a global adversary, which is able to monitor all links and nodes in the network. In order to obfuscate the communication, mix networks use a chain of proxy servers, known as mixes, which cryptographically alter, shuffle, and randomly delay messages before forwarding them to a next hop. It is further assumed, that mixes are under different administrative control and messages "blend" in a number of concurrent messages. Therefore, neither any mix nor a global adversary can link a message to a source and destination at the same time. Both aspects, blending and obfuscating, are the foundation for achieving anonymity. An advanced version of Chaum's original protocol, which addresses many vulnerabilities, is the Mixminion design [61]. Recently, Chaum presented cMix [52], combining the well-known building blocks of mix networks to produce strong anonymity with group-homomorphic encryption for faster message processing. However, due to the random delays, mix networks in general are categorized as *high-latency anonymous communication system* and therefore only support delay-tolerant applications, such as email.

In order to support interactive protocols such as SSH, XMPP, and HTTP, the U.S. Naval Research Laboratory (NRL) developed *onion routing* [93]. Later, it became the core protocol of Tor [72], the standard architecture of *low-latency anonymous communication systems*. To this end, onion routing relaxes the attacker model and accepts the possibility of traffic correlation by trading random delays for interactivity. Since there are no random delays in onion routing, controlling (or observing) the network's edges suffices to correlate individual traffic

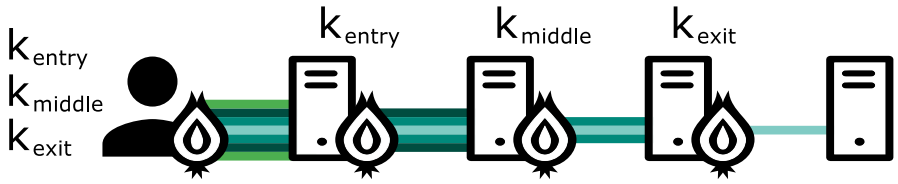


Figure 1: Tor circuit and onion routing.

flows [188]. Indeed, traffic correlation attacks are quite effective [150, 179]. That said, onion routing in general and Tor in particular are not designed to protect from traffic correlation. A global adversary can, in theory, correlate all traffic flows. Even a local adversary with reasonable resources has a realistic chance of deanonymizing a fraction of all Tor users [111, 151]. In practice, though, it seems Tor is still able to protect its users from such powerful attackers [32].

In the remainder of this section we will take a closer look at how Tor implements the onion routing protocol. The Tor network consists of *onion routers*, casually also referred to as relays. Relays are contributed by volunteers and take a similar role as mixes in mix networks. A client who intends to use Tor selects a subset of typically three relays (an entry, a middle, and an exit relay) from a list provided by public directory servers and mirrors. In contrast to mix networks, though, the client will use the selected relays to build a so-called *circuit*: first, the client establishes a symmetric session key k_{entry} with an entry relay by using a variant of the Diffie-Hellman key exchange protocol [169]. Second, by relaying another handshake "through" the entry, the client extends the circuit by one hop, i. e., a middle relay including a respective session key k_{middle}. Repeating this second step and tunneling through all so far chained relays, arbitrary long circuits become possible. However, in order to communicate with non Tor nodes, the last hop must be an exit relay, because all other relays prohibit traffic to exit the Tor network. Eventually, the client's IP address is hidden from the contacted host, who can only see the address of the exit relay. The fact that circuits are built incrementally ensures that relays know their immediate predecessor and successor only. Figure 1 illustrates the principle of a Tor circuit. Tor's protocol specifications, including the handshake protocol, can be found in [71].

When selecting relays for a circuit, there are a few constraints to consider. As we already mentioned, the last relay of a circuit must be an exit relay to connect to destinations outside the Tor network. Destinations inside the Tor network, so-called hidden services, do not require an exit. While relaying exit traffic is sometimes considered to involve a risk of legal actions, Tor introduced exit policies. With exit policies, a relay operator can specify, similar to a stateless firewall, the allowed IP and port ranges or prohibit traffic to exit the Tor network at all. Only if a relay grants exit traffic, it will receive the exit flag and will be considered for the exit position. Besides the exit flag, there are

more flags [194]. Most notably, the guard flag is assigned to relays to tag them as particularly reliable. Guards have been introduced to protect the client from the "prying eyes" of a malicous entry relay. In fact, it has been observed that selecting every new circuit a new entry relay results in a higher probability to select a malicious relay than fixing the first relay [76, 204]. Therefore, Tor clients select a small set of relays with the guard flag as entry and set the lifetime to a random value. The algorithm has been adapted recently [69]; above all, the defaults have been changed from three entry guards to a single entry guard and from a random lifetime of one to two months to a random lifetime of one month to five years.

Since relays are operated by volunteers, the Tor network is highly heterogeneous. In particular, the "donated" bandwidth highly diverges and ranges from 30 kB/s up to 100 MB/s [195]. Therefore, Tor implements a load balancing algorithm. Relays are selected according to a weight published by the Tor directory servers that is basically proportional to the advertised bandwidth of the respective relay. Additional weights exist to balance the different types of relays, e. g., guard and exit relays [194]. Typically, the exit relays' bandwidth constitutes the scarcest resource.

Internally, Tor encapsulates payload data as well as control messages into fixed-sized *cells* to impede traffic correlation. In between a pair of relays, cells travel through TLS-secured TCP connections which are terminated at the relay. Each cell is 512 Byte and consists of a header including a circuit ID. Circuit IDs are assigned on a hop-by-hop basis and thus differ on each segment of a circuit. Upon receiving a relay cell, relays look up the corresponding circuit and depending on the direction of communication add or remove one "skin" of the "onion" encryption, before forwarding the cell to the next hop. That is, if the cell is headed from the client towards the exit, a relay *decrypts* the payload with the session key of the respective circuit. Conversely, if the cell is headed in the opposite direction, a relay *encrypts* the payload. Thus, for the lifetime of a circuit, cells take the same overlay route for both directions.

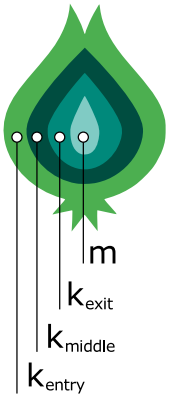


Fig. 2: Layered onion encryption.

Since clients share a key with each relay, they can unwrap the cell and reveal the message m. To create and transmit a cell towards the exit, a client encrypts the message with all session keys one at a time, as depicted in Figure 2, starting with the exit relay's key k_{exit}. The layered encryption ensures that the original message remains hidden as it is forwarded from one relay to the next. Without any additional protection, though, the exit relay is able to read/alter messages. This exit vulnerability of Tor is a privacy risk and facilitates man-in-the-middle attacks [50, 143, 203]. Therefore, it is recommended to employ another layer of encryption, for example TLS, which only the contacted host can decrypt.

Tor supports TCP-based applications only. The Tor client software, the so-called *onion proxy*, acts as a SOCKS proxy [125]. Thus, an application initiates an anonymized TCP connection to a given address and port by contacting the onion proxy. The onion proxy then takes a preemptively constructed circuit and instructs the exit relay to open a TCP connection. It is important to note, that Tor does not tunnel TCP connections, but instead relays application layer data only. The actual TCP connection to the destination originates from an exit relay. Tor conceives a so-called *stream* as internal representation of the data flow associated with the respective TCP connection at an exit. A circuit can carry data from one or more TCP connections and hence can be shared by many streams.

2.4 PERFORMANCE ENHANCEMENTS FOR TOR

Tor's main design challenge is to balance anonymity and performance. Thus, Tor decided to accept a weaker attacker model, which enables traffic correlation and implies the risk of large-scale traffic analysis. The reason for this decision is that protecting from traffic analysis in a low-latency communication system is too expensive, because it would require the implementation of dummy traffic as in [86]. The traffic of many concurrent circuits and streams, though, can also provide the necessary cover traffic to impede traffic analysis of a global adversary [179]. Likely, this is one of the reasons for Tor's success [32]. The authors of [179] also conclude that peer-to-peer approaches such as MorphMix [168] and Crowds [167], where each participant also relays data, weakens the anonymity. Therefore, their results suggest that excessive overprovisioning should not be an option, because otherwise it would facilitate finding "the needle in the haystack".

As we emphasized before, "anonymity loves company" and performance is a driving factor in the efficacy of anonymity networks [70]. Therefore, optimal resource usage while avoiding unnecessary performance penalties must be a central goal of anonymity networks. Otherwise, poor performance can discourage users from using the anonymity service [119], which in turn leads to weaker anonymity. Tor already takes great care to implement efficient cryptographic primitives, i. e., they designed an advanced key exchange protocol [92] and use inexpensive symmetric cryptography for the onion encryption [72].

Unfortunately, this level of attention is lacking when it comes to networking. Tor implements a sliding window mechanism to control the amount of data directed into the network. For every circuit, each edge node, i. e., client and exit, manages a so-called package window that is initialized to 1000 cells. The package window limits the num-

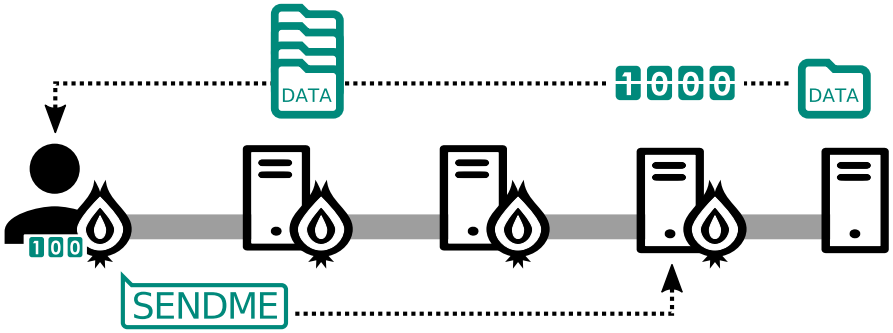


Figure 3: Tor's flow control on circuit level.

ber of cells in flight of any given circuit. A node on the receiving side of a circuit maintains a delivery window, which keeps a record of the number of delivered cells. For every 100 delivered cells, the relay signals to send more data by issuing a circuit-level `SENDME` cell towards the opposite direction. We illustrate this mechanism for a single direction in Figure 3. An additional, analogous mechanism exists on the stream level: the stream-level window's fixed size is 500 cells, and stream-level `SENDME`s worth 50 cells. Due to the sliding window there will be no more than 500 cells in flight on a stream, which is capped by 1000 cells in sum on circuit level. 1000 cells, though, can be a significant amount of data, i. e., approximately 500 kB, so that long queues often build up. In fact, excessive queuing is one of the major causes for huge delays, which Tor experiences painfully [64, 108, 116, 166]. Moreover, long queues give implicit preference to bulk flows which constantly keep the queue filled, when compared to more interactive flows, like for instance web traffic. This motivated a number of circuit prioritization algorithms [22, 110, 189].

The situation becomes apparent when taking a look at Tor Metrics [195], the portal for Tor measurements. It provides statistics about the Tor network and makes the data publicly available. As one part, the torperf tool regularly downloads files over Tor from various vantage points. The median download times of 1 MiB files during the past year are shown in Figure 4. Particularly when considering the quartiles around the median, the results match the daily experience of regular Tor users: largely, the performance is quite reasonable, but for a non-negligible part it significantly deviates and results in high unacceptable download times. A look at the raw data reveals that the time to first byte reaches hundreds of milliseconds up to the order of seconds. While relaying cells multiple times through the Internet inevitably adds additional delays, the results suggest a "clogged" network, which can be attributed to inadequate congestion control. As we will see in the following chapters, Tor's congestion control indeed is one of the main reasons for poor performance, but there are many more contributing factors, which we reveal and discuss.

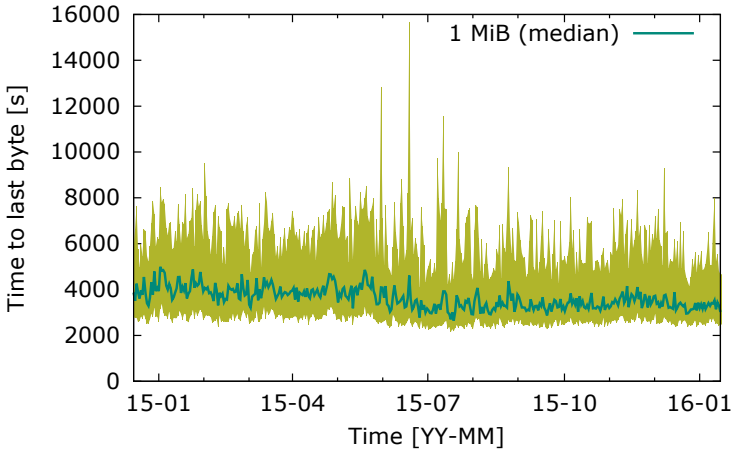


Figure 4: Torperf measurements from Tor Metrics [195].

Since Tor's introduction more than a decade ago [72], low-latency anonymous communication systems have received significant attention in the research community (and beyond). A survey on improvements for Tor can be found in [26]. For obvious reasons, this attention has primarily focused on security and privacy aspects. In the context of our work, results such as [78, 150] are particularly relevant. They selectively put relays under heavy load and observe the effect that traversing circuits experience respective changes in throughput and delay. This can be used to infer the path of a circuit and may lead to deanonymization. It shows the importance of a resilient congestion control algorithm, which should aim for avoiding such interferences as effectively as possible. As we will show at various stages of our work, Tor is exposed to manifold interferences. Our contributions show how to mitigate these effects and make Tor less vulnerable to this attack vector.

In recent years, performance aspects of Internet anonymity in general and the awareness for network congestion issues in particular have become part of the research agenda. Performance enhancements have been proposed, for instance, by considering an alternative circuit selection algorithm [19, 23, 200] or through an adaptive prioritization of circuits [22, 110, 189]. These research directions are orthogonal to our work and remain applicable.

Theoretical models of the Tor network exist, but, so far, only cover anonymity aspects [48, 80, 142]. We on the other hand are interested in resource allocation and fairness aspects, and thus, for the first time, model the Tor network from these perspectives. In particular, we identify gross unfairness in Tor's resource allocation with heavy impact on the performance. The authors of [108, 166] find that cells reside in socket buffers for a long time. In [108], it is suggested to fix this by observing and actively querying all sockets before blindly writing to a socket. Thereby the majority of queued cells is kept in the application

	Transport layer	Reliable transfer	E2E window	HoL blocking	Feedback gap	Circ. unfairness
Tor (vanilla)	TCP	H2H	●	●	●	●
DefenestraTor (N23) [24]	TCP	H2H	○	●	○	●
IPPriv [116, 117]	TCP	E2E	●	○	○	○
PCTCP [25]	TCP	H2H	●	○	●	○
TCP over DTLS [166]	UDP	H2H	●	○	●	○
Torchestra [94]	TCP	H2H	●	●	●	●
UDP-OR [199]	UDP	E2E	●	○	○	○
Unordered TCP [154]	TCP	H2H	●	○	●	●
BackTap (our approach)	UDP	H2H	○	○	○	○

E2E = end to end, H2H = hop by hop, HoL = Head of Line

Table 1: Proposed transport modifications for Tor.

layer, so that scheduling on circuit granularity becomes possible with a smaller backlog between data leaving the application and leaving the host. This follows the general intentions of the approaches in this thesis. However, it does not solve the fundamental problem of excessive standing queues due to circuit windows that often far exceed the bandwidth-delay product—it only moves these queues to a different place. We instead tackle the root cause of the problem.

Transport-related modifications for Tor have been considered before [24, 25, 94, 117, 154, 166, 199]. An incomplete assessment of their feasibility is provided in [149]. Even though each proposal improves individual aspects, most of them still suffer from fundamental performance issues, e. g., they use the same sliding window mechanism and hence inherit the exact same issues as Tor. Table 1 summarizes the characteristics of these approaches and compares them to the original Tor design and our own protocol design. The enumeration of characteristics includes the, from a networking perspective, most relevant aspects and most pressing issues. Throughout this thesis, we will discuss these aspects in great detail and reveal previously unknown issues such as *circuit unfairness* and the *feedback gap*. Below, we will provide a brief summary of related approaches.

As observed by [166], a missing TCP segment carrying data from one circuit will also temporarily stall any other circuit on the same connection until the missing segment has been recovered. TCP does not distinguish between data from different circuits and delivers a

single reliable, in-order bytestream only. This results in *head-of-line blocking* (HoL) upon TCP segment losses.

There are a number of approaches which explicitly [25, 154, 166] and implicitly [117, 199] tackle the the head-of-line blocking. In [154], the authors build upon previous work [153], where they presented an unordered TCP/TLS variant. It enables out-of-order delivery of data and thus does not lead to head-of-line blockings. To this end, the authors extend Tor's cell format and introduce sequence numbers. Except for the unordered delivery, the approach is largely the same as Tor.

The manifest remedy in [166] and [25] is to use separate (loss-based) TCP connections per circuit. As consequence, a lost segment stalls only a single circuit. Both approaches are conceptionally identical and for most parts differ in implementation details only. In [166], it is envisaged to implement user-space TCP per circuit and tunnel the segments over UDP/DTLS. PCTCP [25] varies the idea by using Kernel TCP per circuit and shielding it with an IPsec tunnel between adjacent relays. However, such a modification does not overcome the fundamental problems with Tor's window mechanism and the corresponding feedback gap. That is, it focuses only on congestion control of individual, isolated overlay hops, but does not improve the end-to-end (E2E) feedback along the entire circuit. As we point out in this work, it would also largely increase the (already very high) aggressiveness of the traffic, due to the higher number of parallel loss-based TCP connections.

Torchestra [94] aims to mitigate the effect of bulk flows. Their approach is to separate bulk traffic from interactive traffic and to multiplex it into two different TCP connections, one for each type of traffic. They argue that circuit prioritization algorithms are limited as they can only affect the scheduling in the Tor application itself. However, Torchestra neither solves the fundamental congestion problems nor the head-of-line blocking. In addition, they also increase the aggressiveness of the overall Tor traffic.

Only [24, 117, 199] get rid of the fixed-size window. UDP-OR [199] builds upon UDP to tunnel a TCP connection through the entire circuit, i. e., the TCP connection starts at the client's application and is terminated at the exit relay. Thus, the exit still establishes the actual TCP connection to the destination. IPPriv [117] takes it one step further and establishes TCP connections, as with non-anonymized communication, between the client's application and the destination. The circuit consists of nested IPsec tunnels, which alter IP addresses and serve the same purpose as Tor's layered encryption. Since both approaches leave it to the client's application to use reliable communication, they naturally support non-TCP based applications. However, while this may be considered a very clean design, it soon reaches its limits because of the long round trip times of a full circuit, which im-

pairs the responsiveness of both reliability and congestion control [29]. Additionally, using complex protocols like TCP end-to-end also come at a significant risk of leaking identifying attributes [185], so that end-to-end designs are generally not favorable [25].

The DefenestraTor approach [24] substitutes Tor's fixed-size end-to-end window by a hop-by-hop (H2H) window, with a scheme adapted from congestion control in ATM networks, namely N23 [120]. With N23, each relay along the circuit is assigned an initial credit balance of $N2 + N3$ cells. Relays decrement the credit by one for each forwarded cell. Every N2 forwarded cells, relays send a flow control cell downstream, which induce an increment of the credit accordingly. Apart from these modifications Tor remains untouched. Hence, head-of-line blockings and circuit unfairness remain open issues. Moreover, the question regarding the choice of suitable window parameters cannot be answered generally (very much for the same reasons as for Tor's fixed-size window).

Our brief enumeration clearly demonstrates that the various intertwined mechanisms in a Tor-like overlay are complex and easily start interfering. The interferences, though, have not been considered before and demand an integral perspective. At the same time, all previously discussed approaches unanimously emphasize the call for action regarding Tor's data transport. While virtually all transport-related approaches continue to use standard TCP with its built-in congestion control, we develop a tailored transport design for anonymity overlays, which eliminates the need for an end-to-end window and also solves many other issues.

Beside Tor, the other low-latency anonymity overlays deployed and used today, namely JonDonym [37] and I2P [104], suffer from comparable performance problems, mostly due to various transport layer effects [171, 201]. Likewise, proposals such as Crowds [167] and Tarzan [86] seem prone to similar congestion issues. Therefore, although focusing mainly on the Tor network, our findings are to some extent also applicable to the performance issues of other anonymity overlays.

2.5 RELATED PERFORMANCE ISSUES

Beyond specific improvements of Tor, there is existing literature on transport aspects of overlay networks in general. Work on multicast overlays, as for example in [28, 30, 122, 197], is mainly driven by the objective of distributing data flows along a tree of subscribers and the congestion control challenges, which arise in such scenarios. The load of message flooding and query processing in peer-to-peer networks is the object of research in [100, 118]. However, they consider short

message bursts only and not active data streams as they occur in anonymity overlays.

A conceivable design avenue, which we also take, is to employ an alternative transport protocol instead of TCP. Many novel and tailored protocols have been proposed in various contexts [82, 85, 98, 187, 192]. QUIC [98] for example is optimized for HTTP/2 semantics. The main improvements include instant connection establishment and stream multiplexing without head-of-line blocking. With respect to congestion control they heavily borrow from TCP CUBIC [95].

As another example, BitTorrent [56] introduced a transport protocol named µTP [187]. µTP implements congestion control based on the "less-than best-effort" principle [180]. It aims to be "over-friendly" to TCP, i. e., to give way for TCP connections when sharing a bottleneck. Both, QUIC and µTP build upon UDP and demonstrate the promising direction of such an approach. However, these protocols do not take the specific architecture of anonymity overlays into account. The design space of anonymity overlays is much larger and much more complex. If applied in Tor, QUIC and µTP would exhibit similar problems as Tor currently does, because they would not be able to provide congestion feedback along an entire circuit.

The general design of a Tor circuit resembles a Split TCP setting as it also occurs in performance-enhancing proxies (PEPs): data is forwarded from an incoming to an outgoing TCP connection, linked by an application-layer queue. A survey on PEPs, including case studies, can be found in [42]. Split TCP was originally developed in the context of wireless and satellite communication, but nowadays also finds use in content distribution networks [160]. It basically subdivides an end-to-end TCP connection into a sequence of typically two concatenated connections, where a middlebox (e. g., a wireless access point or a router) acts as a PEP.

By terminating the connection at the middlebox and acknowledging data before the actual destination received it, Split TCP, in fact, violates TCP's end-to-end semantics. If desired, this can be avoided by acknowledging data upstream only after it has been acknowledged by the downstream node [205]. In the context of anonymity networks, such a strict adherence to TCP semantics is generally considered unnecessary, though (just like for most practically deployed PEPs). In a network like Tor, a cell loss further downstream implies a broken circuit, which will be considered by the source as a reason to switch to a fresh circuit anyway. Since Split TCP aims for maximizing the utilization of link capacities, PEPs buffer data and hence congestion might become a problem. As it has been noted before [137], using Split TCP in an overlay network poses particular challenges in this and many other regards. Therefore, even though we focus on the case of anonymity networks, some of our results may also be applied in the area of PEPs and for other overlay designs.

3 HOW (NOT) TO BUILD A TRANSPORT LAYER

3.1 OVERVIEW

This chapter is based on previous work by the author [7, 8] and collaborative work [2, 3]. The general idea presented in Sec. 3.3 should be attributed to fellow co-authors.

Overlay protocol designers, including those of anonymity overlays, tend to perceive overlay links as an equivalent of dedicated point-to-point links, just like the ones forming the basis of the Internet. Another instance of data forwarding and transport functionality is then added on top of the existing Internet protocol layers. The respective protocol mechanisms are part of the application and thus of the application layer protocol. There, it is tempting to take up concepts from their Internet counterparts. An unreflected re-use is treacherous, though: as we will demonstrate, the fact that relays are end systems can induce interferences between transport layer connections. There are more such effects and constraints which are often and easily overlooked.

Therefore, it is highly necessary to think about anonymity overlays from a network performance perspective: while there is a significant body of work on security and privacy aspects, there are surprisingly little insights into how to design the overlay in such a way that it makes efficient and proper use of network resources. Previous work in the area of anonymity overlay performance typically focused on isolated aspects, and the proposed mechanisms more often than not cause undesired deterioration in other parts of the system.

This chapter explores the intrinsically broader design space. By doing this we create awareness regarding the specifics of anonymity overlays. We particularly stress that the behavior of the interacting underlay and overlay transport layers is complex and often causes unexpected—and unexplored—side effects. Therefore, we show that so far considered combinations of overlay and underlay protocols cannot deliver good throughput, latency, and fairness at the same time, and we establish guidelines for a future, better suited transport layer design.

3.2 DESIGN SPACE

In anonymity overlays unicast connections are forwarded over a sequence of overlay links. These overlay links are transport layer connections through the Internet. They can be thought as virtual point-to-point connections which form the overlay network. We illustrate this in the bottom two layers of Figure 5. In analogy to the terminology of circuit-switched networks (and also resembling Tor's terminology),

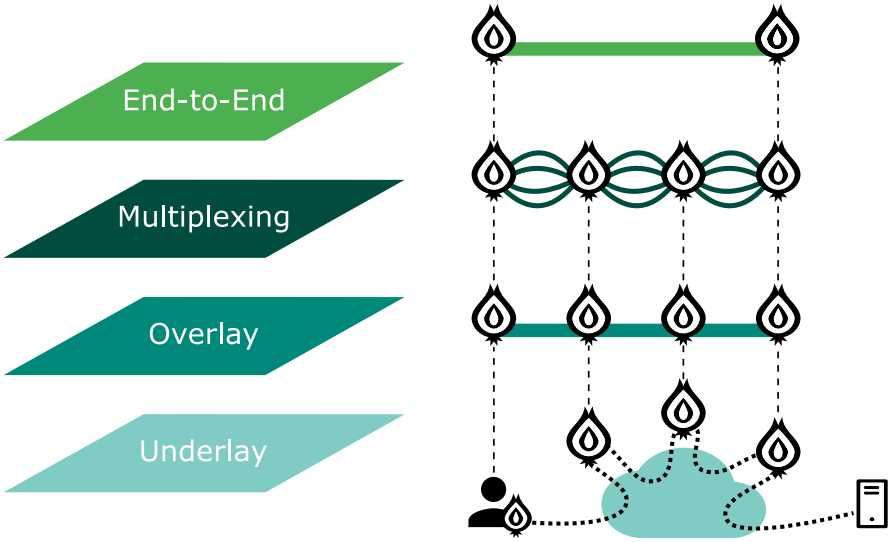


Figure 5: Layers in the transport architecture design space.

we call the end-to-end unicast tunnels circuits. While on the Internet, end-to-end means from source to (true) destination, that is, from client to server, we take the perspective from the Tor network and refer to the client (i. e., the onion proxy) and the exit relay as the respective endpoints. In the following we will take great care that these nuances in terminology should become clear from the context.

The most obvious design choice is the transport protocol used for hop-by-hop communication, i. e., for the overlay links. Overlay links may, in principle, be based on TCP, UDP, or any other protocol implemented on its endpoints. In practice, only TCP and UDP are widely available. Both can be extended to provide security and message integrity by SSL/TLS or DTLS, respectively. Likewise, IPsec tunnels between neighboring overlay nodes can provide an additional layer of encryption. To avoid the necessity of implementing a transport protocol in the operating system kernel, alternative transport layers are sometimes realized in user space and tunneled over UDP. Another reason for the same measure can be to not use overly many transport layer sockets. As it has been shown in [90], for example, socket exhaustion can easily become an attack vector in Tor. Realizing alternative transport layers in the user space is possible since UDP—apart from port addressing—does not change the service model of IP. An example are QUIC [98] and BitTorrent's µTP [187]. In the context of Tor Reardon et al. propose an architecture which tunnels application-layer TCP connections over UDP [166].

A single overlay link can be traversed by a varying number of circuits. The most commonly used architecture—and arguably the one closest to the "idea" of an overlay—is that all circuits traversing a given overlay link share one common transport layer connection. However, it is also conceivable to establish multiple parallel transport

layer connections [25, 94, 166]. As shown in Figure 5, this degree of freedom can be perceived as another layer in the design space, which we call the *multiplexing layer*: here, circuits are multiplexed into one or more transport layer connections.

One of the most central aspects in the choice of the transport protocol is an appropriate congestion control mechanism. The choice for a transport protocol on each overlay link implies a choice for a congestion control mechanism used on that link. There is a broad spectrum of options: TCP alone comes in a variety of different flavors [15]. Further possibilities range from an application-specific mechanism (as in QUIC and µTP) or no congestion control at all in case of UDP.

The latter need not necessarily be a bad idea, if appropriate congestion control is performed on higher layers, e. g., on the circuit level. Indeed, it is noteworthy that per-link congestion control alone does not suffice anyway: consider a circuit which first traverses a high-bandwidth link, and then subsequently a much tighter one. If there were no feedback, data would pile up before the bottleneck, and would either need to be dropped or lead to excessive queues. Therefore, regardless whether a design implements congestion control on individual overlay links, circuit-level congestion feedback is always necessary. In Tor, for example, TCP is used on each link, complemented by a circuit-level end-to-end sliding window.

Note that none of these concepts limit whether the traffic that is carried *through* a circuit is UDP-like datagram or TCP-like bytestream traffic. Given proper encapsulation, any overlay design can carry both. The only noteworthy difference is that pure UDP-like traffic allows for additional degrees of freedom, since reliability and order guarantees need not be provided then.

In all cases, one needs to be aware of the implications on anonymity. In general, end-to-end approaches come at a higher risk of leaking information, as headers and parameters are forwarded in an unmodified way along the whole circuit. This is also the reason why active queue management techniques (like RED [84] or ECN [165]) do not appear to be a wise choice within an anonymity overlay. On a general level, hop-by-hop feedback seems more appropriate, because it does not directly reveal parameter choices or other information to further away nodes, and because it can make use of per-circuit knowledge that anyway exists in intermediate nodes.

3.3 INTERFERENCE BETWEEN OVERLAY CONNECTIONS

In various contexts, it has already been shown that end-to-end Internet connections may not be mistaken as point-to-point links. One well-known example is the *TCP meltdown* effect: if a TCP connection is sent through a TCP-based VPN tunnel, the stacked TCP implementations will start interacting. This can cost 55 % or more throughput

performance [115]. In this section we demonstrate that TCP-based overlay applications can also experience TCP interactions. We argue that such effects should be taken into account in the design of overlay networks.

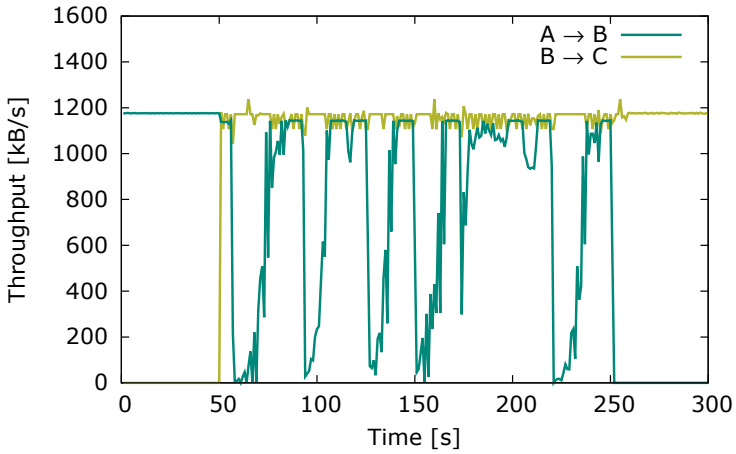
Overlay nodes are end systems. Therefore, all their TCP connections to all their communication partners share one physical link. The overlay nodes' end system character has significant—and non-obvious—impact on the overlay network's performance.

To illustrate the effect we are dealing with, we perform an experiment with real network nodes. Three Linux (kernel 2.6.26, TCP CUBIC) hosts A, B, and C are connected via an Ethernet switch. To resemble typical Internet links, we configured all Ethernet interfaces to 10 Mb/s full-duplex. Two TCP connections are set up, from A to B and from B to C. Just like two overlay links from the same end system, both connections share the link between B and the switch. For the first 250 seconds, A continuously transmits bulk data to B. After 50 seconds, B starts a bulk data transfer to C, also for 250 seconds.

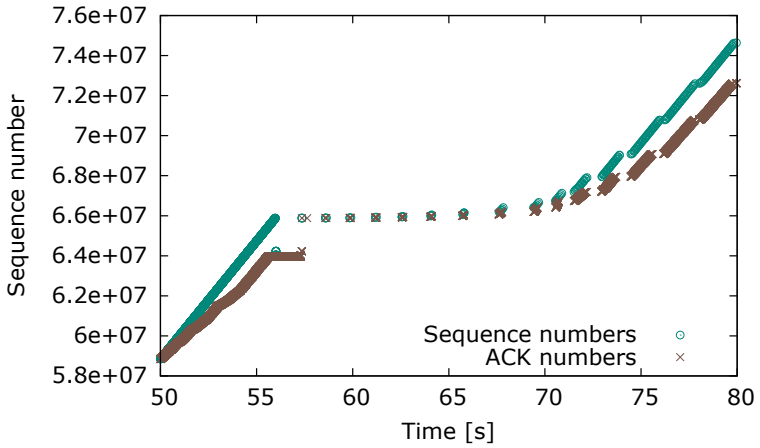
Figure 6a shows the application-layer throughput over time in one single experiment run. During the first and last 50 seconds, when only one of the connections is active, the throughput is, just as one would expect, close to 10 Mb/s. However, when both transfers are active, the picture is drastically different. The problematic connection is the incoming one: A → B does not make full use of the bandwidth, but instead oscillates heavily.

The reason lies in the outgoing queue of node B's link. In this queue, there are (a) TCP data segments for B → C and (b) `ACK`s for A → B. When B → C sets in, the respective queue increases drastically in length, resulting in an increased queuing delay. This delay also affects the acknowledgments for A → B. The effect is clearly visible if we take a detailed look at the sequence number progression for A → B during one of the oscillation cycles, shown in Figure 6b. It visualizes the sequence numbers of outgoing data segments and incoming `ACK` segments at node A over time, for a 30 seconds time interval immediately after B → C starts to transfer data.

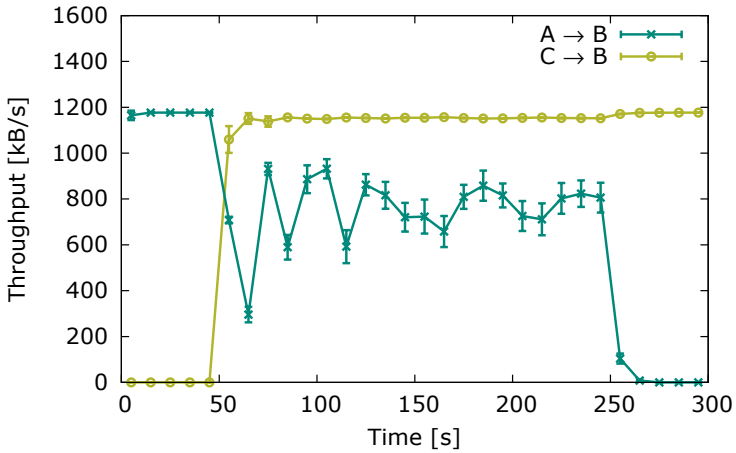
The RTT increases rapidly as the outgoing queue at B grows. This can be seen from the increasing horizontal gap between outgoing data segments and incoming `ACK`s between seconds 50 and 55. Around second 55.5, a segment loss occurs. Because of the long queue it takes a very long time until this loss is recovered. A is ultimately forced into a slow start. The transfer during the slow start is not continuous, but `ACK`s arrive (and new segments are released) in "batches". This matches the "ACK compression" effect described in the early 1990s [147, 207, 208]. This happens over and over again, and causes the oscillations, which in turn result in a significant loss of throughput. To obtain a statistically sound results, we performed 100 independent experiment runs. Figure 6c shows the results in 10 seconds



(a) Throughput in a single experiment run.



(b) Sequence number and ACK progression of A → B.



(c) Mean throughput over 100 simulation runs.

Figure 6: TCP interactions on overlay links.

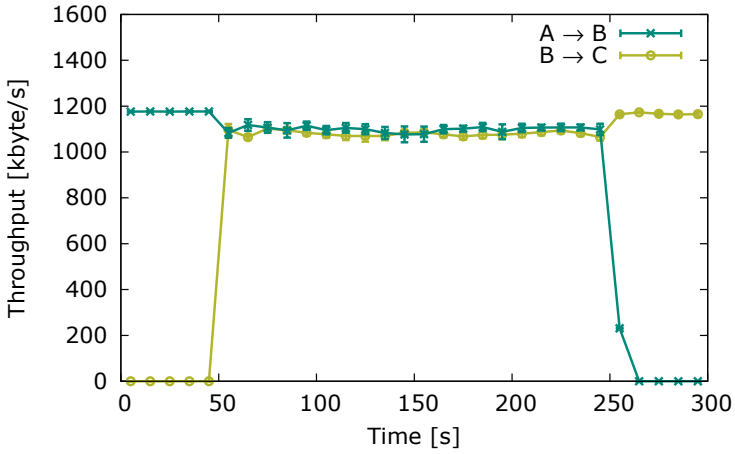


Figure 7: Impact of ACK prioritization.

intervals as an average over all runs with 90 % confidence intervals. The mean throughput from A to B decreased by about one third as soon as the transmission from B to C starts.

3.3.1 *Traffic Shaping and ACK Prioritization*

So far, the effects that we demonstrated have not been taken into consideration in the design of overlay networks. However, related phenomena are known in the context of TCP performance on asymmetric links (e. g., ADSL Internet access links) [31]. A possible remedy is to prioritize TCP ACK segments in the outgoing traffic stream by using traffic shaping mechanisms [31, 113]. This technique is an important first step towards solving the throughput oscillation problem. In this section, we therefore take it up and adapt it to our needs.

The key idea is easy to understand: ACKs are forwarded to the interface with higher priority; this avoids that the ACKs suffer from long delays due to many large data segments from other connections. Such mechanisms are actually not uncommon in practice. They can be configured using the standard Linux kernel's class-based queuing features, similar means exist in other operating systems. In fact, there are software packages to optimize the performance of asymmetric Internet connections, like [103]. Even some access points provide corresponding features. We have set up such an ACK prioritization mechanism in host B in our experimental setting above and assessed its performance impact. Figure 7 shows the results obtained with this modification, as an average over 100 runs with 90 % confidence intervals. In this simple scenario, the problem is apparently solved.

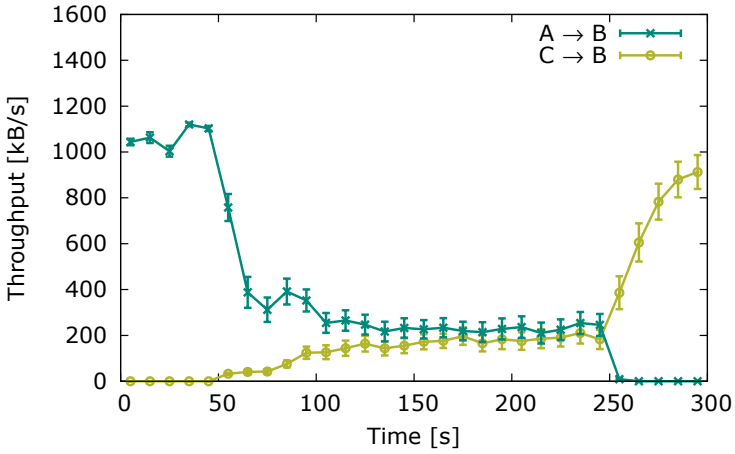


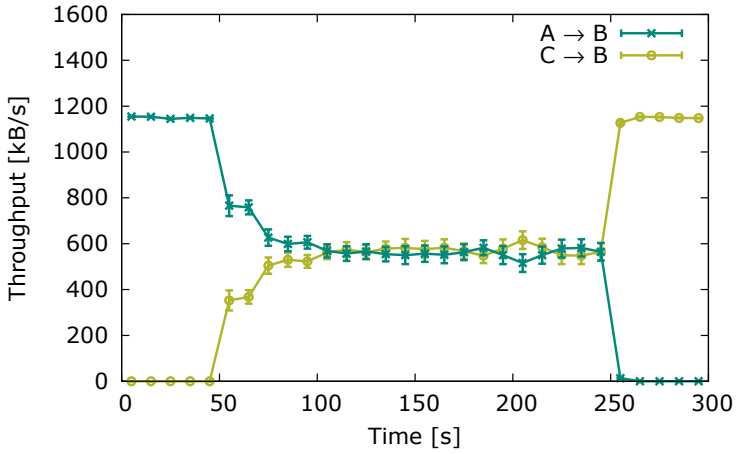
Figure 8: Bidirectional communication without ACK prioritization.

3.3.2 *Bidirectionally Used TCP Connections*

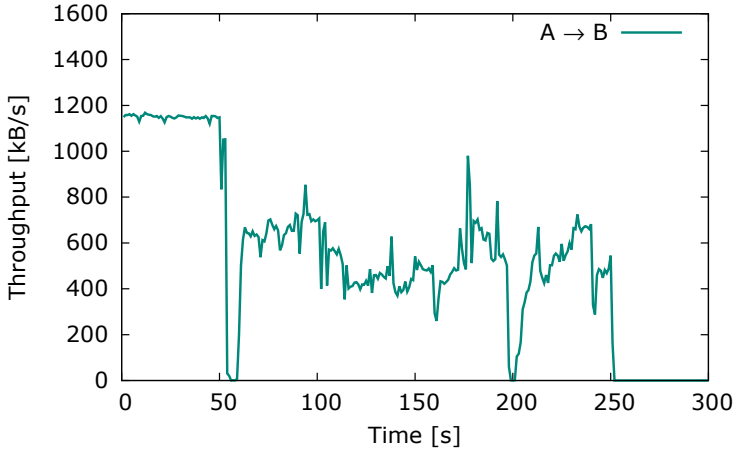
In many overlay networks, TCP connections are used bidirectionally. For instance in Tor, many circuits are multiplexed over a single overlay link that can easily lead to bidirectional traffic. In order to asses this magnitude, we designed a metric in cooperation with the Tor project and deployed it [131]: in 10 s intervals, relays record for each active overlay connection (i. e., pushed more than 20 KiB) the number of read and written bytes. If the fraction of bytes in one direction is at least 10 times the other direction, the connection is classified as either "mostly reading" or "mostly writing". All other active connections are classified as "both reading and writing", that is, bidirectional use. The statistics reveal, in the median approximately 50 % of all connections are used bidirectionally [195]. In fact, this feature distinguishes typical overlay communication from most client-server protocols.

In case of bidirectional communication, TCP uses piggybacked acknowledgments. Then, however, ACKs cannot be separated and preferred as described in the previous section. Thus, in case of bidirectional TCP traffic with piggybacked acknowledgments, we should not expect ACK prioritization alone to perform as well as it did for unidirectional traffic.

To see what happens in the case of bidirectional traffic, we have again performed experiments in the same setting as above. Host B still communicates with hosts A and C over one TCP connection each, but now both connections transfer data in both directions simultaneously. Again, the communication between A and B starts at second 0 and ends after 250 seconds, and the connection between B and C sets in after 50 seconds and ends at second 300. Figure 8 shows the achieved throughput over time for the bidirectional data streams



(a) Multi-run averages.

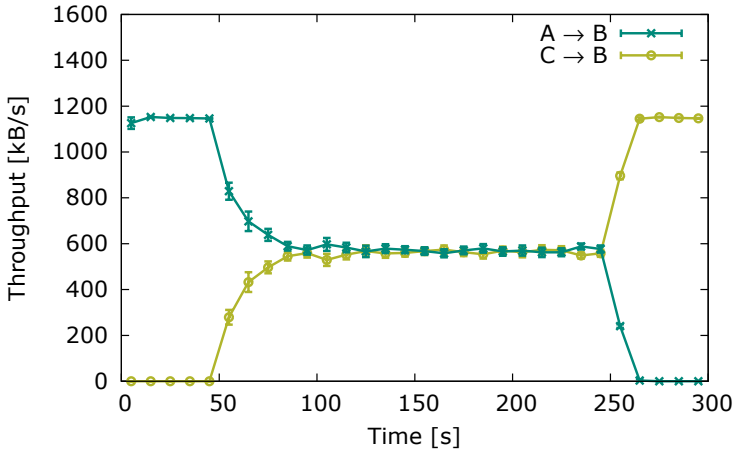


(b) Individual connection ($A \to B$).

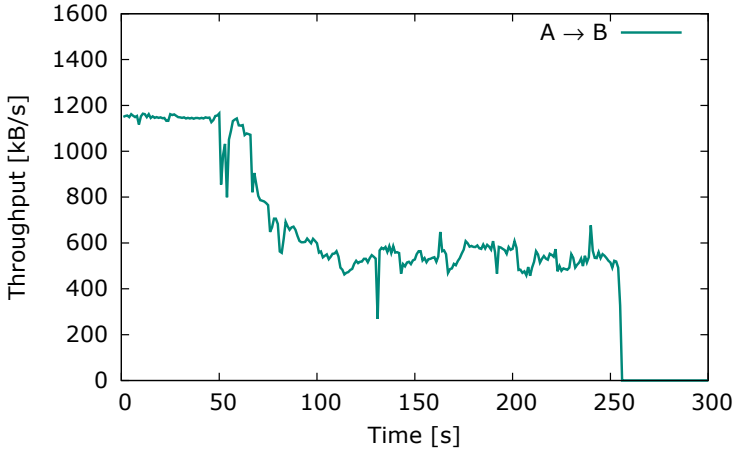
Figure 9: Bidirectional communication with ACK prioritization.

without ACK prioritization. We show the throughput from A to B and from C to B—as we have seen before, the incoming data transfer towards B is the problematic direction. In the figure, we see a massive decrease in throughput to around 200 kB/s when both connections are active.

If we repeat the experiment with the ACK prioritization mechanisms described above, the result are surprising at a first glance: despite piggybacked ACKs, the throughput deterioration virtually vanishes. Both connections now make full use of the available bandwidth (note that each direction of B's link is now shared between two data transfers). This is shown in Figure 9a. The reason for this improvement becomes clear upon closer examination. There is, in fact, a small number of non-piggybacked acknowledgments. They are prioritized and, upon their arrival, serve as cumulative ACKs. This apparently suffices to remedy the throughput decrease. A more detailed look, however, reveals that this remedy is treacherous: the mean throughput increases, but



(a) Multi-run averages.



(b) Individual connection (A → B).

Figure 10: Bidirectional communication with separate TCP connections.

strong oscillations remain. This is not evident in the multi-run averages in Figure 9a, but upon examination of individual connections' behavior the still unsteady behavior becomes apparent. We show one typical example in Figure 9b.

Clearly, prioritizing the small number of individual `ACK`s that occur despite bidirectional traffic is not enough. However, piggybacked `ACK`s cannot be prioritized as described above, because they cannot easily be separated from the data packets. We therefore argue that overlay designs should take this into account. If bidirectional data exchange between the same pair of peers is required, a simple solution is to separate incoming and outgoing traffic into two different TCP connections.

We have verified the efficacy of this proposal in another experiment. Now, two TCP connections are set up between hosts A and B, and another two connections between B and C. Each of these connections is used in one direction; transmissions start and end as in the previ-

Approach	Standard deviation
w/o ACK prio	208.4 kB/s
w/ ACK prio	181.2 kB/s
w/ ACK prio and sep. conns	100.6 kB/s

Table 2: Throughput standard deviation (SD).

ous experiments. Apparently, piggybacked acknowledgments will no longer exist in this setting. Figure 10a shows the throughput of the data streams towards B over time. Figure 10b is a throughput-over-time plot for one single representative connection. Here, we observe a significantly more stable throughput.

Note that by this mechanism we do not deteriorate TCP fairness: TCP congestion control works independently for both directions, so TCP fairness properties are not altered.

To underline that the observed improvements are indeed statistically significant, we also performed a different evaluation of our experimental results. To this end, we consider all throughput samples for data flows towards B from all 10 s time slots between seconds 100 and 250 from all experiment runs. For this set of samples, we calculate the standard deviation. A high deviation means that the throughput samples are spread over a large range of values, a low deviation means that the throughput is generally more balanced. The results, shown in Table 2, corroborate and quantify what the individual connection plots in Figures 9b and 10b already indicate: `ACK` prioritization brings with it a clear benefit, but only with the per-direction separation of TCP connections can undesirable effects be avoided with full effectiveness.

3.3.3 *Implications*

In the design of overlay networks, the transport layer has mostly been seen as a black box; this point of view is definitely not enough. Undesired interactions between overlay links may incur performance penalties and unexpected effects. We thus argue that researchers and practitioners must take the transport-layer implications of their overlay designs into consideration.

Based on real-world network experiments, we analyzed the problem and discussed how to overcome it in the context of existing overlay networks. Fortunately, relatively simple, minimally intrusive, and readily deployable strategies yield significant benefits. However, as we will see in the remainder, deploying separate connections might not always be viable, because the number of concurrent connections has an impact on the performance too. Thus, for overlays with many concurrent connections the aim must be to keep the queues as short as possible to mitigate the effects in the first place.

3.4 THROUGHPUT, LOSS, AND DELAY

3.4.1 *Analytical Throughput Characterization*

The achieved throughput is perhaps what first comes to mind when the aim is to "improve the performance" of an anonymity overlay. Not surprisingly, many of the existing works focus on improving throughput performance by more or less subtle modifications. Their effects are typically demonstrated either by measurements on real hard- and software, or by simulations.

More general insights can be obtained by taking analytical characterizations of transport protocol performance into account. While the authors of [88] and [164] use the less sophisticated TCP model by Floyd and Fall [83], which for example does not consider TCP implementation parameters, we build upon the TCP performance model by Padhye et al. [157]. In the remainder of this chapter, we will go beyond using the TCP model to derive throughput and use it to reveal fundamental relationships to other parameters, like the expected packet loss frequency.

TCP is by far the most important transport protocol, and virtually all anonymity overlay designs make use of TCP in one way or another. TCP performs congestion control based on packet loss: when transmitted segments do not arrive at the receiver, this is taken as an indication of network congestion. Hence the source node reduces its window size (and thereby indirectly its sending rate). Since TCP implements reliable transport, lost segments are retransmitted. Assume that the number b of packets that are typically acknowledged in a single `ACK`, the retransmission timeout (RTO) T_0, and the round trip time (RTT) of the network path are given. If we furthermore assume that there is no hard limit on the window size (i. e., TCP is free to make full use of the available bandwidth), Padhye et al. give the following formula for TCP Reno, which sets the packet loss frequency p observed by the TCP connection and the achieved throughput B into relation:

$$B \approx \frac{1}{\mathrm{RTT} \cdot \sqrt{\frac{2bp}{3}} + T_0 \cdot \min\left\{1, 3\sqrt{\frac{3bp}{8}}\right\} \cdot p \cdot (1 + 32p^2)}. \qquad (1)$$

Typical values for the parameters are $b = 2$ (cf. delayed `ACK`s) and $T_0 = 0.2\,\mathrm{s}$ (cf. initial RTO).

In a nutshell, a higher RTT and/or a higher loss rate p will increase the denominator of equation (1), and thus—all other parameters unchanged—corresponds to a lower achieved throughput. Note that the available bandwidth of the used network links does *not* appear in the formula: if the available bandwidth is exceeded, packet drops will occur (i. e., p will increase), which in turn implies a lower throughput.

Padhye et al.'s model describes the *steady state*, in which bandwidth and observed packet loss have reached an equilibrium.

This also underlines that throughput and loss are closely intertwined, so that they cannot be treated independently. While some of the more modern, post-Reno TCP variants (NewReno [101], CUBIC [95], etc.) vary this theme in different regards, this is still at the heart of how TCP congestion control typically works. In fact, it is likewise the case for other congestion controlled transport protocols on the Internet: if they aim to be *TCP friendly*, they must react at least as strongly to packet loss as TCP, so as not to unfairly "steal" bandwidth from competing TCP connections.

Delay-based congestion control like, for instance, in TCP Vegas [44] or µTP [187], takes changes in the RTT into account in order to react early to congestion and to thus reduce the number of packet drops. Yet, this comes at the cost of a high sensitivity to changes in the RTT, as they are to be expected on long, bandwidth-scarce overlay paths. For TCP Vegas, a discussion can be found in [123]. It also means that such protocols react earlier than classical TCP variants if they compete for bandwidth at a common bottleneck—and that they therefore "lose" in such a competition.

3.4.2 *Multiplexing on TCP Overlay Links*

Perhaps the most important implication of the interrelation between throughput and packet loss becomes clear if we look at the multiplexing layer. Which differences should we expect if we either use (a) one single connection between each pair of relays shared by all circuits traversing the overlay link, or (b) multiple independent TCP connections, each carrying a single circuit?

For simplicity, assume that no other connections are present on an overlay link's underlay path and that n circuits are currently active on the overlay link. Then, in case of (a), one TCP connection with bandwidth B will carry all circuits. For (b), there are n connections, each with bandwidth B/n. We saw above that a lower bandwidth corresponds to a higher packet loss frequency. Therefore, if $n > 1$ connections share a bottleneck, each of them will *necessarily* experience a higher packet loss rate than one single connection across the same TCP bottleneck.

We show this effect in Figure 11, where we "invert" Padhye et al.'s model: while (1) cannot be solved for p in closed form, we can use Newton's method to determine the value of p for which the predicted throughput reaches a given level. In the figure, we vary the number n of circuits, which are either (as, e. g., in Tor) multiplexed over a single connection or (as proposed in TCP-over-DTLS [166] and PCTCP [25]) use separate, parallel connections. Here, we chose a total bandwidth of 1 Mb/s and an RTT of 200 ms. We see that the number of multi-

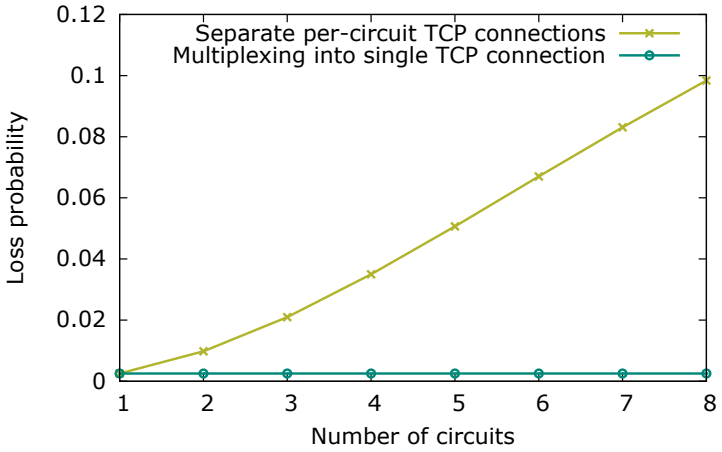


Figure 11: Loss probability according to Padhye model.

plexed circuits into a single TCP connection does not affect the loss probability. The picture looks completely different for multiple parallel TCP connections, where the loss probability increases drastically. Packet-level simulations with ns-3 [102] confirm the results.

Note that this is an inherent property of TCP congestion control. It does not depend on where exactly the congestion control mechanism is implemented, be it at the transport layer or in userspace tunneled over packet-based transport.

Of course, TCP is a reliable transport protocol, therefore losses will be repaired by retransmissions. Nevertheless, packet loss is a problem, because it causes additional delay: until the loss is repaired, no subsequent data can be forwarded to the application. We will soon look at this head-of-line blocking problem in more detail. For now, we conclude that a higher number of TCP connections on an overlay link very significantly increases the number of packet loss events.

3.4.3 *Transitional Effects*

While the Padhye model only covers the steady state, transitional effects also need to be considered. This becomes clear when we contrast the alternatives of using TCP on the end-to-end level in combination with datagram transport on individual overlay links on one hand, and per-hop TCP connections on the other hand. The former is, for instance, used in IPPriv [116] and UDP-OR [199].

There are two effects that should be taken into account: first, we observe that the former approach implies separate TCP connections per end-to-end connection. Therefore, if multiple circuits share one overlay link, there necessarily are multiple parallel TCP connections. Consequently, similar observations as above hold: the bandwidth per TCP connection will be lower, and a higher frequency of packet losses is to be expected. This is particularly problematic in a setting with

long TCP connections spanning the whole path of a circuit, because packet retransmissions for repairing these losses will take place end-to-end through the overlay. Due to the longer RTT, lost packets will take longer to be detected and to be repaired.

Second, the high RTT of an end-to-end path through an anonymity overlay has implications on TCP dynamics before the steady state is reached: classical TCP variants will take much longer to initially ramp up the bandwidth. In order to get an idea of the time span for this ramp-up, we performed network simulations with ns-3 [102], packet-level network simulator. We implemented an overlay in ns-3, which we modeled closely after Tor. We used TCP NewReno, which is today (beside CUBIC) the most common TCP implementation. The scenario is intentionally kept very simple, in order to clearly show the key reasons for the performance problems that will, of course, likewise be present in more complex setups. Circuits with fixed-window end-to-end congestion control traverse a sequence of three intermediate overlay nodes (= relays). All up- and downstream connections of a given overlay node share a common link to the network core.

The RTT between any pair of nodes through the underlay is set to 80 ms. All links are configured to 10 Mb/s, except for one bottleneck link (between the intermediate overlay nodes closest to the client) which is restricted to 1 Mb/s. Ideally, the throughput achieved by the circuit would be 1 Mb/s. In practice, protocol overheads and the effects of congestion control will not allow to always fully utilize the bandwidth.

In Figure 12, we show the throughput of a single circuit over time, measured end-to-end on the application layer. The hop-by-hop TCP line shows the result with separate TCP connections along each overlay hop. End-to-end TCP denotes one single TCP connection along the whole circuit, forwarded over datagram transport. As can be seen, the latter takes time in the order of one minute to reach a throughput level that is comparable to what hop-by-hop TCP can deliver almost instantaneously.

TCP CUBIC ramps up based on real-time clock ticks and thus overcomes the RTT dependency. However, as shown in [127], CUBIC generally—that is, also for short RTTs—converges to the long-term bandwidth rather slowly. Therefore, our conclusion from these results is that end-to-end TCP (or TCP-like) congestion control reacts far too slowly to constitute a viable design alternative.

3.4.4 *Delay*

Because anonymity networks forward data multiples times over potentially long Internet links, higher latency compared to a direct connection is inevitable. The overlay nodes are located all around the world, so that the sum of delays on the traversed overlay links can

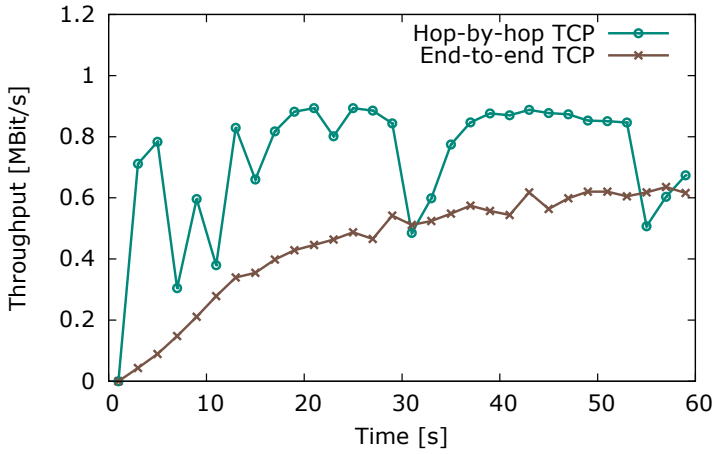


Figure 12: Circuit throughput in ns-3 simulations.

easily be hundreds of milliseconds. The question is: how much *additional* delay is caused by the transport layer design?

Additional delays occur when data is queued for processing or forwarding. An interesting observation is that if multiple TCP connections are used in parallel, then queuing delays on the IP layer will typically be higher. This can be understood by again looking at Padhye's TCP model: a higher number of parallel TCP connections through a common bottleneck will, as discussed before, result in a higher packet loss rate. That is, queue overflows—the source of packet loss—happen more frequently. Consequently, higher packet loss rates also correspond to queues that are, on average, longer.

We performed ns-3 simulations similar to those described above, and increased the number of circuits. We found that the queue lengths are indeed 10% higher than for a single TCP connection, with correspondingly higher queuing delays.

For reliable transport protocols like TCP, packet loss is another source of delays: lost packets need to be retransmitted. As observed by Reardon et al. [166], this has severe implications if multiple circuits are multiplexed over one transport layer connection. TCP delivers one single reliable, in-order bytestream and does not distinguish between data from different circuits. Consequently, a missing segment with data from one circuit will also temporarily stall any other circuit on the same connection. This is called the head-of-line blocking problem.

Reardon et al.'s remedy is to use separate TCP connections (implemented in user space and encapsulated into UDP). Likewise, PCTCP by AlSabah et al. [25] use per-circuit TCP connections to tackle the problem. As seen above, though, this comes at the cost of an increased loss rate for each individual connection. To assess the impact, we build upon our results from Section 3.4.2 above and determine the frequency at which loss events would affect any individual circuit (assuming TCP segments of 1,500 byte). The results are shown in

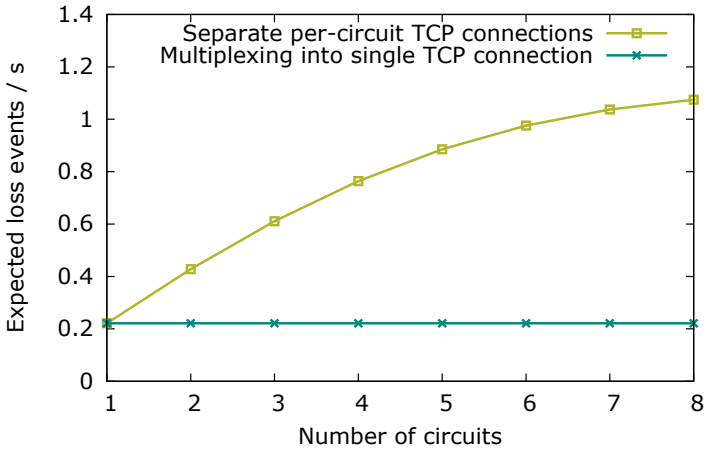


Figure 13: Loss event frequency per circuit.

Figure 13. The additional delays in fact *overcompensate* the gain that results from separating circuits into independent connections: each circuits suffers *much more often* from segment recovery delays.

Apart from network layer queues and the need to wait for retransmissions, application layer queues in the overlay implementation can also cause significant delays. For example, as we have shown in [4, 12], significant latency in Tor is caused by undesired interactions of two independent rate limiting mechanisms and their large refill intervals.

3.4.5 *Fairness Towards Other Applications*

The number of active connections has an impact on the fairness towards other applications. This is easy to see from a straightforward gedankenexperiment: assume an anonymity overlay using n parallel connections across a network link (e. g., the Internet link of an overlay node), and a separate, independent application which also has a TCP connection open on that link. For comparable RTTs at least, TCP connections share bandwidth fairly. Thus, since there are $n+1$ connections in total, n of which belong to the anonymity overlay, this overlay will get a total share of $n/(n+1)$ of the bandwidth, while the other application gets only $1/(n+1)$.

A higher number of (TCP-based or TCP-friendly) transport layer connections will thus, if considered in sum, be more aggressive. Proposals which increase the number of TCP connections therefore are at risk to unfairly disadvantage other applications running in parallel to the anonymity overlay. Thus, we need to keep an eye on the number of parallel TCP connections and keep them on a low level if maintaining the fairness to other applications is one of the goals.

On the other hand, unequal sharing of the bandwidth need not even necessarily be wrong or undesired—but it should be independent from the number of active circuits and overlay links (i. e., n). Ide-

ally, it should be configurable by the user. This can even be considered an additional incentive for donating bandwidth: overlay node operators might be more willing to allocate additional bandwidth if they are able to configure how aggressively this bandwidth is "claimed" by the overlay if other applications also have demand.

3.5 LESSONS LEARNED

In summary, what have we learned about the choice of transport protocols when building anonymity overlays? Can we narrow down the design space?

First, we conclude that pure end-to-end mechanisms without reliability and congestion control on individual overlay links are not a favorable design choice. The high total latency along an entire path through an anonymity overlay will necessarily lead to very long reaction times for repairing packet loss and to slow convergence towards a balanced bandwidth share. The high number of connections causes unfairness towards other applications. This is particularly undesirable since the degree of unfairness heavily depends on the current load situation in the overlay; it is thus outside the sphere of influence of the node operator.

Separate per-hop, per-circuit, loss-based connections are likewise not a convincing solution: the aggressive behavior of a large number of bundled TCP connections results in (again, non-controllable) unfairness towards other applications and in frequent packet loss. The latter, in turn, causes frequent delays while waiting for retransmissions. Multiplexing circuits into shared connections, on the other hand, has the disadvantage of head-of-line blockings. As we will see in Chapter 6, this approach also raises fairness issues.

From these insights it already becomes clear that standard congestion control approaches are insufficient. Consequently, we argue that it is necessary to design a tailored protocol. In the following chapters, we will underline our assertion and pursue the idea of an overlay-aware congestion control protocol. According to the arguments in this chapter, such a protocol must satisfy a number of requirements: first of all it is necessary to take both the underlay *and* the overlay into account, i. e., make use of knowledge about individual circuits and thus of the specifics of anonymity overlays. While the protocol operates on circuit granularity to avoid head-of-line blockings, it must be scalable. That implies handling many concurrent circuits without excessive socket resource consumption and without being overly aggressive. Likewise, we aim for short queues to reduce delays.

Such a scheme could well be implemented in the application/multiplexing layer over UDP transport, without modifications to the overlay nodes' operating system. We believe an effective approach would

be to observe RTT variations along outgoing overlay links and to continuously adjust the assigned sending window.

Of course, as discussed before, this needs to be combined with some form of end-to-end congestion feedback to avoid excessive queuing. Again, it seems wise to make use of the fact that intermediate overlay nodes already maintain per-circuit status information: nodes in an anonymity overlay can (and should) actively contribute to per-circuit congestion control by appropriate feedback and queue management. In this thesis, we show that a hop-by-hop, backpressure-based congestion control scheme that actively handles per-circuit backpressure between neighboring hops is a solution to the performance problems in anonymity overlays.

3.6 CHAPTER SUMMARY

In the design of anonymity overlays, the transport layer has attracted limited attention so far. However, it has a tremendous impact on various aspects of the system's performance. Wrong design choices incur huge performance penalties. We therefore believe that it deserves much more consideration.

In this chapter, we explored the design space for transport layers in anonymity overlays. In doing so, we revealed a number of common misconceptions and mistakes, which led to the conclusion that existing approaches cannot meet the (performance) requirements of anonymity overlays. From our insights though, we derived guidelines of a transport layer design. In the following chapters, we bring this vision to reality.

4

PRIVACY-PRESERVING NETWORK MEASUREMENTS

4.1 OVERVIEW

This chapter is based on previous work by the author [5, 6].

In this chapter, we propose a privacy-preserving method to determine the number of distinct users, who connected to an Internet service. The key motivation for our work are the problems Tor is facing when it comes to estimating the number of users [96, 132, 134]. The technical challenges of guessing the number of users stem from the fact that anonymity networks are inherently good at hiding. Nevertheless, it would still be very interesting to obtain statistics about the total number of users of the system, as such data provides valuable insight on the magnitude of the anonymity set and supports future development of the system, e. g., with parameter choices. Tor node operators, though, should not exchange (or even record) explicit information about locally observed IP addresses of Tor users. Thus, the challenge is to determine the total number of *distinct* IP addresses that have used Tor, while avoiding to exchange information about observed sets of user IP addresses between Tor node operators.

Beyond Tor and its challenge to estimate the number of active users, many other types of statistics and applications are faced with similar challenges, e. g., location privacy, web analytics, and health care statistics. Privacy demands are obviously just as essential in these contexts as in the case of Tor, and our algorithmic means can be applied there as well.

Thus from a more general perspective, we are looking for a way to obtain the number of distinct elements in a multiset of user IDs (e. g., in Tor the IP addresses of users). These user identifiers can occur multiple times at the same service entry point (e. g., in Tor the directory servers and mirrors), and they can occur at multiple entry points. No single point, however, is able to see the multiset as a whole. We then need to exchange and combine the information in a privacy-preserving way. In particular, an honest-but-curious adversary should not be able to conclude with high probability that a specific user has indeed been active in the system by looking at the information exchanged between node operators.

In order to tackle this challenge, we start from an existing algorithm for probabilistic counting [81]. When applied naively, we show that in the worst case it is still possible to conclude with arbitrarily high probability that a specific user was present. We thus proceed

with an improved design which avoids that an attacker can gain such knowledge.

In our analysis, we use the relative knowledge gain of an attacker as a privacy metric, and measure it by comparing the attacker's a-priori knowledge (before taking the information from the exchanged data structures into consideration) and a-posteriori knowledge (that is, the level of confidence when the attacker makes use of the exchanged data). This novel, information-theoretic viewpoint allows us to appropriately capture the degree of privacy obtained by our algorithm. Based on this methodology, we will demonstrate that even in the worst case the knowledge gain of an attacker is limited.

Our considerations and the algorithmic design are based on probabilities and random decisions. However, it should clearly be noted that this does not mean that user privacy is guaranteed only with a certain probability. Quite in contrast: since the attacker does not know which random decisions have been taken by the algorithm, there remains significant uncertainty on the side of the attacker. And in the end, preserving privacy is always a question of maintaining some uncertainty on the side of the observer.

The key contributions include (a) a privacy-preserving duplicate insensitive counting algorithm, (b) an in-depth privacy analysis based on an information-theoretic methodology which focuses on the attacker's probabilistic knowledge gain, (c) an accuracy analysis which assesses the impact of all parameters, and (d) an exchange mechanism to collect and derive results from distributed service points while revealing minimal information to all participating entities.

4.2 USER COUNTING IN TOR

Since Tor's primary aim is to protect the anonymity and privacy of its users, collecting information about the network and its usage is very challenging [134]. In order to set up anonymized connections, Tor clients request a list of relays from directory mirrors. Therefore, directory mirrors are a kind of service entry point to the Tor network and are the ideal data source for user statistics. Based on access statistics gathered by these directory mirrors, Tor Metrics presents daily usage statistics of Tor as described in [59, 132]. Since raw user information should not be exchanged between mirrors for privacy reasons, these statistics are based on the number of requests occurring at directory mirrors. However, only a small fraction of all users will contact a particular mirror, and it is not known how many distinct mirrors an average Tor user contacts per day. The results are therefore coarse estimates at best, because they depend on both the probability of a user selecting one specific mirror and the total number of directory requests generated by an average user per day—and both can only be guessed.

The huge uncertainties associated with this approach have been pointed out before in [96]. There, it was envisaged to count the number of observed IP addresses at multiple directory mirrors instead. Yet, no concrete solution was proposed for how to combine records from multiple mirrors to obtain accurate results while preserving privacy. For example, simply exchanging and summing up the counter readings of all mirrors would result in an gross overestimation due to users contacting multiple mirrors. On the other hand, exchanging lists of observed IPs would thwart privacy.

In contrast to all previous approaches, our algorithm allows to collect information from multiple directory mirrors in such a way that the total number of *distinct* IP addresses across all participating mirrors can be obtained and user privacy is preserved.

4.3 ALGORITHMIC BASIS

In this section, we start by introducing the algorithmic basis of our approach—*FM sketches*, introduced in [81] by Flajolet and Martin. They are an algorithmic mean to estimate the cardinality of a multiset: for a multiset M of n elements, an FM sketch can determine the approximate number of distinct elements in M. While finding the exact number of distinct elements in a multiset of size n requires $O(n \log n)$ time, FM sketches only need $\Theta(n)$ steps to find a good estimate. This trait was the original motivation behind them. However, as we will see, they also exhibit other very interesting properties, which make them a worthwhile basis for solving the problem considered here. We therefore recapitulate the key ideas behind FM sketches in the following; more details can be found in [81].

An FM sketch is based on a bit vector $S = (s_1, \ldots, s_w)$, $w \geqslant 1$, which is initialized to zero. It also requires a hash function h_1 with geometrically distributed positive integer output, where the probability that $h_1(x) = j$ (with $j \geqslant 1$) for any randomly picked element x equals $P(h_1(x) = j) = 2^{-j}$.

When an FM sketch is used to estimate the cardinality of a multiset M, each element $x \in M$ is hashed using h_1. Each hash value is interpreted as an index in S, and the corresponding bit $s_{h_1(x)}$ is set to one. This leads to a bit pattern in S, which, due to the geometric distribution of h_1, will typically have many 1-bits on the left and many 0-bits on the right. This is outlined in Figure 14, where four distinct user IDs $ID_A, \ldots, ID_D$ are hashed into a sketch.

Flajolet and Martin found that a good estimate for the number of distinct elements can be obtained from the length of the uninterrupted, initial sequence of ones in S, i. e., from

$$Z := \min \left\{ j \in \mathbb{N}_0 \mid s_{j+1} = 0 \right\}.$$

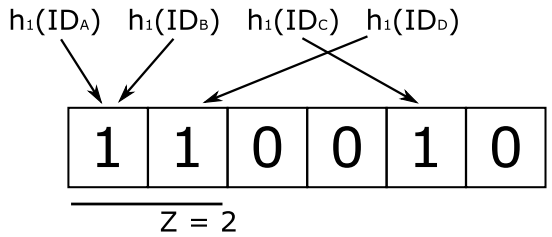


Figure 14: An FM sketch: IDs are hashed with a geometrically distributed hash function and bits are set respectively. By evaluating the initial sequence of set bits Z it is possible to obtain an estimation of the distinct number of users.

There is a constant factor $\varphi \approx 0.77351$ based on which an estimate C for the number of distinct elements inserted into the FM sketch can be obtained as

$$C = \frac{2^Z}{\varphi}. \tag{2}$$

Since the derivation of this constant will become relevant for our algorithm design discussed later on, let us see how φ can be obtained. The probability that bit j is not set after n distinct elements have been added is

$$P\left(s_j = 0|n\right) = \left(1 - 2^{-j}\right)^n$$

The probability $P(z \mid n)$ that *at least* the first z bits in a sketch row are set to one after n additions is therefore

$$P(z \mid n) = \prod_{j=1}^{z} \left[1 - \left(1 - 2^{-j}\right)^n\right]. \tag{3}$$

The probability for *exactly* z initial ones is $P(z \mid n) - P(z+1 \mid n)$. This allows to obtain the expected value of Z given n as follows

$$E[Z \mid n] \quad = \quad \sum_{z=1}^{w} z \cdot (P(z|n) - P(z+1|n)).$$

We can combine this with equation (2) to get an expression for φ depending on n:

$$\varphi\,(n) = \frac{2^{E[Z|n]}}{n}. \tag{4}$$

The dependency on n can be overcome: $\varphi(n)$ goes to $\varphi \approx 0.77351$ for $n \to \infty$. It converges so quickly that the value can be considered constant in practice. A numerically easy way to obtain the value of φ is to evaluate (4) for sufficiently large n (e. g., $n = 10^5$).

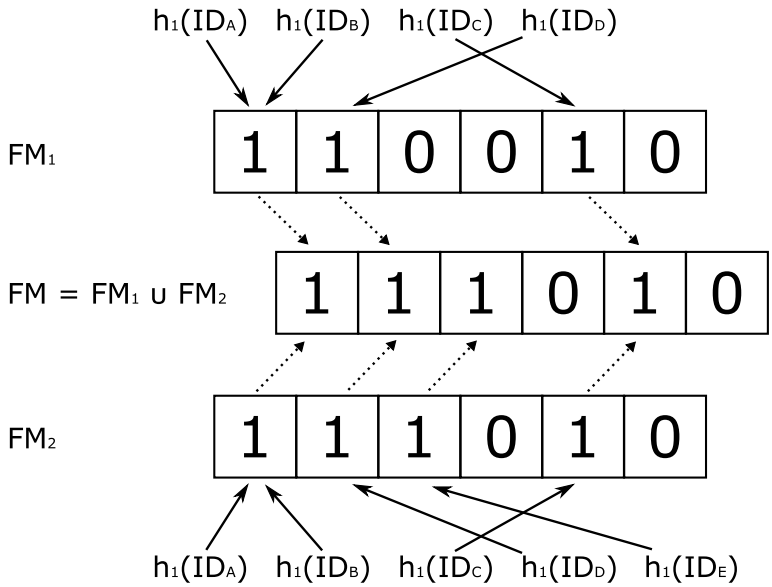


Figure 15: Basic FM sketch setting: combining sketches with a bit-wise OR results in a sketch which represents the union of both individual sketches.

The accuracy of the estimation can be improved by using multiple sketches in parallel. The respective technique is called Probabilistic Counting with Stochastic Averaging (PCSA) in [81]. With PCSA, each element is first mapped to one of the sketches by using a uniformly distributed hash function h_2, and is then added to this (and only this) sketch. In the remainder of this chapter, we will use the PCSA variant.

If m sketches are used with PCSA, then the estimate for the total number of distinct items added is given by

$$C = \frac{m \cdot 2^{\sum_{i=1}^{m} Z_i / m}}{\varphi}, \tag{5}$$

where Z_i is the number of leading 1-bits in the i-th sketch. One can identify a PCSA set with an $m \times w$ matrix, where each of the m rows is a standard FM sketch. Upon addition of an element, a uniformly distributed hash function h_2 selects one row, and a second, geometrically distributed hash function h_1 picks one column. The thereby selected bit is then set to one. For a sufficiently large number of elements, PCSA yields a standard error of approximately $0.78/\sqrt{m}$ [81]. Increasing m thus results in a higher estimation accuracy. Note that increasing m also increases the total number of 1-bits in the PCSA matrix for a given, fixed multiset; this trait will have an impact on the privacy vs. accuracy tradeoff later on. For very small element counts in the order of m or below, there are well-known initial inaccuracies. These can be partially alleviated by switching to a different evaluation method, based on "hit counting" [202], when Z is small. For details, we would like to refer the reader to the respective discussion in [129].

The width w of the matrix is not critical: only the left hand side of the matrix is relevant for the estimate, and (as can be seen from (2)) the width of the relevant block of ones increases only logarithmically with the element count. Therefore, it is easy to set w to a sufficiently large value. We use $w = 64$ here, which will generally suffice for most practical applications.

Multiple FM sketches (and likewise PCSA matrices) can be merged to obtain the total number of distinct elements added to at least one of them by a simple bit-wise `OR` as depicted in Figure 15. Observe that combining the FM sketch with all elements of a multiset A and the FM sketch with all elements of another, possibly overlapping multiset B using bit-wise `OR` produces an FM sketch that is identical to the sketch of multiset $A \cup B$. Elements present in both A and B will not be counted twice, since the respective bit will always have value 1 in both sketches. This duplicate insensitivity trait paves the ground for distributed user counting.

4.4 NAIVE DISTRIBUTED COUNTING

Now let us look at how we could naively apply FM sketches to distributed user counting. We will subsequently come to talk about the drawbacks and remaining problems, and how they can be overcome. Each service entry point maintains a PCSA matrix with preconfigured dimensions $m \times w$. m and w, as well as the hash functions h_1, h_2, are agreed on by the service operators in advance. When a user with ID u contacts the service at one of the entry points, u is hashed into the sketch matrix and the respective bit is set locally. Clearly, if the same user contacts the entry point more than once, the user will not be counted again, since the respective bit is already set. By evaluating the sketch from a single entry point, we therefore obtain an estimate for the number of distinct users who contacted the respective mirror.

We assume that at the end of an observation period the sketch matrices from multiple service entry points are collected and merged. We will discuss later on how this collection can be accomplished. The bitwise logical `OR` of the individual entry points' sketches results in a sketch for the total number of *distinct* users contacting *at least one* of the respective service entry points. Note that the merged sketch is equivalent to the one obtained if all users connected to one single entry point. The users are therefore counted in a duplicate insensitive way.

In the specific use case of Tor, the counting operation could be performed at the directory mirrors. As already argued in [96], this is a reasonable design, because each Tor user contacts at least one mirror during bootstrapping.

4.4.1 *Adversary Model*

In an application which collects and combines statistics from multiple independent sources, a malicious adversary can easily manipulate the results by adding false information. For instance, in a distributed user counting mechanisms, the attacker may pretend to have seen many additional IP addresses. An attacker can trivially do so even in very simple schemes without means for privacy protection (e. g., exchanging lists of IP addresses). Adding privacy protection will not make it harder to mount such manipulation attacks (and does, in fact, also not aim to do so). Therefore, analyzing such an attacker in the context of privacy-preserving distributed statistics does not yield new insights.

Much more interesting is the question how much information an attacker can gain by observing the exchanged data. The following privacy analysis therefore considers an adversary model based on honest-but-curious observers or participants, i. e., users who are willing to honestly contribute their information but who will try to find out whatever they can about other users.

In particular, we assume that the attacker has access to the bit matrix that results from the bitwise logical `OR` of all service entry points' sketches. As we will see, it can be achieved that no instance in the system obtains more information than this. Hence, no instance in the system will be able to see any individual entry points' sketches apart from their own local one. What we would like to prevent is that an adversary looking at the combined sketch is able to conclude that a particular user has been active and used the service. On the other hand—and here is the tradeoff—sketches should still be able to produce accurate results in a duplicate insensitive and distributed way.

4.4.2 *Privacy Analysis Methodology*

If sketches are applied as discussed above, user IDs are not exchanged directly. Typically, many different user IDs are mapped to the same bit in the sketch matrix. Therefore one might expect that the sketch does not reveal much information about which specific users contacted the service. This conclusion is treacherous, though. Let us first discuss this intuitively, before we come to a more formal treatment.

Recall that hash function h_1, which selects the column in the sketch, is geometrically distributed. Consequently, the lion's share of all IPs are mapped to the first bits. Therefore, an attacker cannot learn much from the observation that such a bit is set: it could have been set by one (or many) out of a huge number of potential users. This cannot be our measure, though, because we want to protect the privacy of *all* users, and not just of those who happen to be mapped to a far-left bit in the sketch. And if an attacker observes that a bit far to the right

is set, then it becomes suddenly very likely that a specific user has indeed been present in the system.

In the extreme case, an attacker knows for sure that only one single out of all possible user IDs maps to a specific location in the sketch; if this bit is set to one, the attacker can be sure that the user has contacted the service. This is not at all an unrealistic setting: for instance, in the case of IPv4 addresses, the attacker may easily be able to brute-force calculate the hash positions of all possible IP addresses. And he may ultimately find that only one single IP address maps to a given bit.

In order to prepare for our subsequent extensions and their analysis, we go beyond these trivial cases and explore the relationships and implications more formally and in more detail. We then use the obtained insights as a basis for an improved design which will overcome the problems. In particular, we consider, in more generality, the question how "sure" an attacker can be that a specific user has contacted the service. This is expressed by the *a-posteriori probability* of the user being there, after the information from the sketch is known. A probability of one means that the user has contacted the service for sure, a probability of, e. g., 0.5 means that it is equally likely that the user has or has not been there.

Clearly, this a-posteriori probability for the presence of a given user depends on the attacker's *a-priori knowledge*: how certain has the attacker been about the user being active *before* taking the information in the sketch into account? If the attacker, for whatever reason, has been 99.9% sure that the user has been active before even looking at the sketch, then the a-posteriori probability will in any case be high, too: at least 99.9%, as the attacker will not "lose" information by looking at the sketch. We will therefore always look at the a-posteriori knowledge *depending on the attacker's a-priori knowledge*. The a-priori and a-posteriori probabilities of a user being active provide a well-defined and formally tractable basis for rigorous analysis; we therefore use these probabilities to express the adversary's a-priori and a-posteriori knowledge. The smaller the difference between the a-priori and a-posteriori probabilities, the less information an attacker can gain. We can therefore use this as an information-theoretic metric for an attacker's information gain, and thus for the degree to which the privacy of a user is protected.

In contrast to previous analyses of privacy-preserving algorithms, we are thereby able to quantify the information gain for arbitrary probabilistic a-priori knowledge. This viewpoint allows both a generic and intuitive discussion of tradeoffs and design choices. Nevertheless, it comes with some small pitfalls. In particular, we will see that even in the final version of our algorithm, there will always be a certain knowledge gain (in the above sense) for an attacker. This general effect, however, is per se inevitable if usage information in

any form is exchanged: even just revealing the total number of users already provides "some" evidence for an attacker. If it is, for instance, known that a large number of users out of a limited set of potential users used a service, this alone already can already increase the attacker's confidence in the presence of a specific user, and thus the attacker's a-posteriori knowledge. Our aim can thus only be to effectively limit the possible knowledge gain based on the exchanged information, while still allowing to obtain duplicate-insensitive statistics across operator boundaries. A key benefit of our analysis methodology is that it is, in a certain sense, honest and transparent: it clearly reveals and quantifies also the effects of an attacker becoming just slightly more confident, even if very significant uncertainty remains.

4.4.3 *Privacy Analysis of Naive Sketch-Based User Counting*

Our aim here is to design a distributed user counting algorithm in such a way that the possible knowledge gain of an adversary is as small as possible. In order to analyze the situation in case of a naive application of FM sketches to distributed user counting, we consider the situation where a specific user ID u has indeed used the service, along with a total of n users. Let i and j respectively denote the row and column IDs in the sketch to which u maps (i. e., $i = h_2(u), j = h_1(u)$). Since we assume that u has used the service, it is clear that bit (i,j) in the sketch will be set to one. We also assume that the attacker knows the total number of users n; obtaining this information at least approximately will not be difficult—in the end, making this information available is the purpose of our scheme. The attacker does not know for sure whether user u has used the service, but has a certain level of "suspicion" before looking at the sketch. This a-priori knowledge of the attacker is expressed by the a-priori probability $P(u)$. The cases where the attacker has definite a-priori knowledge ($P(u) = 0$ or $P(u) = 1$) are trivial, so we assume that $0 < P(u) < 1$.

We are now interested in the probability $P(u \mid (i,j))$ that the user has used the service after the attacker has learned that bit (i,j) is set. This probability will depend on $P(u)$, but also on the position of the bit in the sketch and on the matrix dimensions: as argued intuitively above already, bits on the right hand side of the sketch are "used" by a lower number of different user IDs and will therefore reveal more information if they are set. To this end, first observe that due to the distributions of h_1 and h_2, a user ID maps to the FM sketch matrix position (i,j) with probability

$$\frac{1}{m \cdot 2^j},$$

since one out of m rows is uniformly chosen and, independently, a column is picked with geometric distribution. Consequently, if n dis-

tinct user IDs have been set at random, the probability that the bit at position (i,j) has been "hit" at least once is

$$1 - \left(1 - \frac{1}{m \cdot 2^j}\right)^n.$$

To determine the a-posteriori probability $P(u \mid (i,j))$, we use Bayes' theorem [35]. In order to apply it, we need the probability at which an attacker would expect bit (i,j) to be set, given that n users have contacted the service *and* given the a-priori knowledge $P(u)$. This probability is given by

$$P((i,j)) = 1 - \left(1 - \frac{1}{m \cdot 2^j}\right)^n \cdot (1 - P(u)),$$

because the bit is one if it is either set by user u (with a-priori probability $P(u)$), or by at least one out of n other users (each with probability $1/(m \cdot 2^j)$), or by both. Now, according to Bayes' theorem, the a-posteriori probability $P(u \mid (i,j))$ is

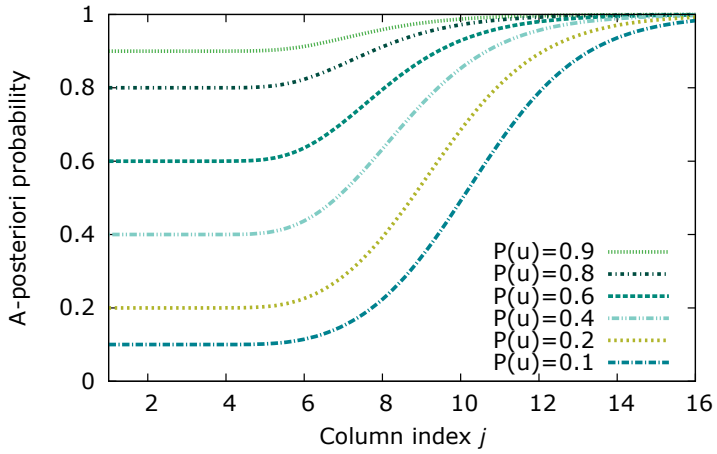
$$P(u \mid (i,j)) \quad = \quad \frac{P((i,j) \mid u) \cdot P(u)}{P((i,j))}.$$

Here, $P((i,j) \mid u)$ denotes the probability that position (i,j) is set if user u has been present; clearly, by the construction of the FM sketches, this probability is equal to one. Thus

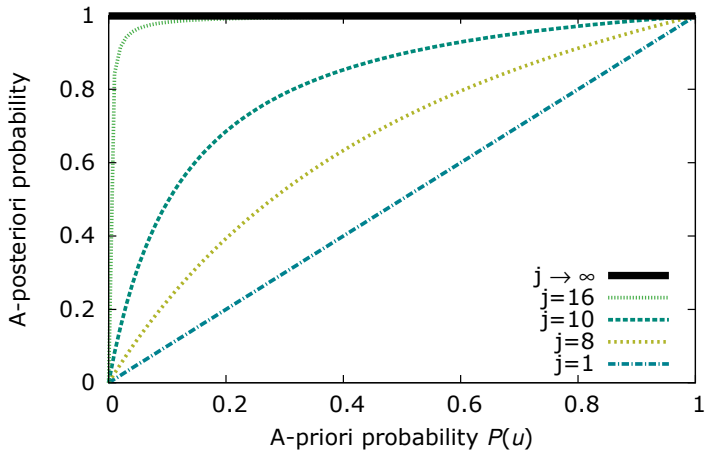
$$P(u \mid (i,j)) \quad = \quad \frac{P(u)}{1 - \left(1 - \frac{1}{m \cdot 2^j}\right)^n \cdot (1 - P(u))}. \tag{6}$$

One can see that the a-posteriori probability depends on several parameters, including j, n, m, and $P(u)$. In particular, the equation supports our intuition that bits to the right, to which a lower number of user IDs are mapped, are more problematic: the higher the column index j, the more information an attacker gains if the bit is actually set. This relationship is depicted in Figure 16a, for a PCSA sketch using $m = 8$ rows, where $n = 1000$ users have contacted the service. The figure shows the a-posteriori probability according to (6) for varying j, where the individual lines correspond to varying a-priori knowledge $P(u)$.

Clearly, if the user ID under consideration is mapped to one of the first few columns (in the specific case here, up to about $j = 5$), an attacker does not gain significant knowledge. This is good news for most users, as the large majority of user IDs will be mapped to one of the first few bits, i. e., in case of Figure 16a this will be 97% ($\sum_{j=0}^{5} 1/2^j$) of all users. However, after $j = 6$, the a-posteriori rapidly approaches one. So, if a user is unlucky and u is mapped to a bit far to the right, an attacker can determine that the user has contacted the service with very high confidence.



(a) Impact of j on the a-posteriori knowledge: the higher the column index j, the more information an attacker gains if the bit is actually set.



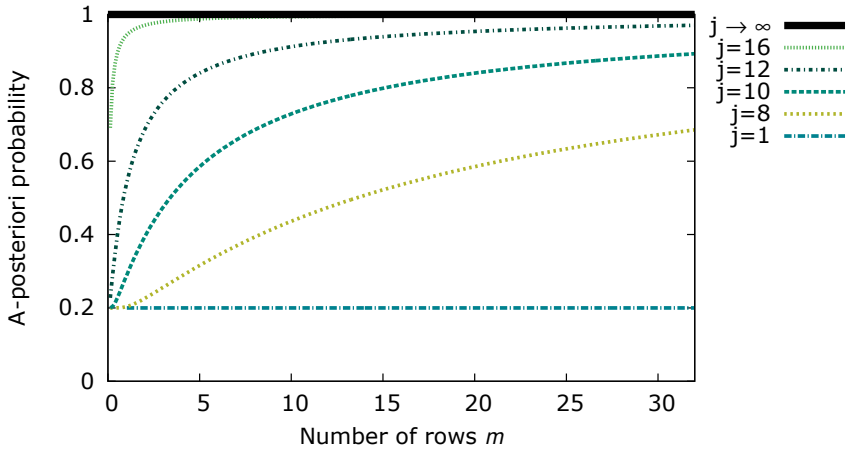
(b) Knowledge gain: in the worst case, an attacker may become arbitrarily sure about a user's presence.

Figure 16: Privacy evaluation ($m = 8$, $n = 1000$).

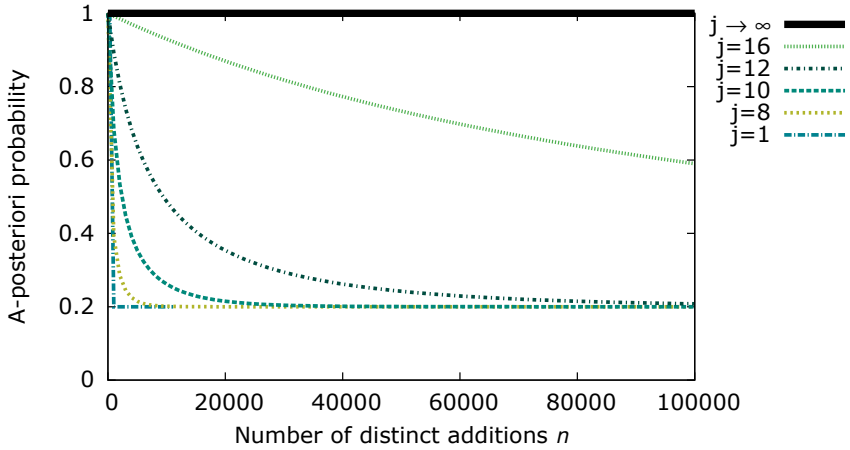
This problem is confirmed in Figure 16b, which shows the a-priori probability on the x- and the a-posteriori probability on the y-axis. For $j = 1$, we get an almost ideal result where an attacker does not learn much by examining the sketch: a-priori and a-posteriori knowledge are almost equal. But for increasing j, the attacker can gain more and more knowledge.

Since our aim must be to protect the privacy of *all* users, we have to take the worst case into account. This can be done by taking the limit of the a-posteriori probability for $j \to \infty$. This limit turns out to be equal to one for any $P(u) > 0$:

$$\lim_{j \to \infty} P(u \mid (i, j)) = 1.$$



(a) Impact of increasing m (for $n = 1000$): increasing m leads to better a-posteriori knowledge for the attacker, but also increases the accuracy of our estimation.



(b) Impact of increasing n (for $m = 8$): a higher number of users n yields higher uncertainty for an attacker, but it does not affect the amount of knowledge gained in the worst case.

Figure 17: Impact of varying parameters for $P(u) = 0.2$.

Thus, even without any significant a-priori knowledge, an attacker may become arbitrarily sure about u's presence in the worst case. The limit is shown as a bold line in Figure 16b and subsequent figures.

Note that $j \to \infty$ is equivalent to the case discussed in the beginning of this section, where the attacker was able to narrow down the set of IDs and therefore knows that only one single possible user ID maps to a certain bit. These considerations do not depend on the size of the ID space and thus hold generally.

The impact of m and n is also interesting to assess. Increasing m leads to better a-posteriori knowledge for the attacker (cf. Figure 17a), but also increases the accuracy of our estimates as discussed above. This results in an inherent tradeoff of accuracy versus privacy. A higher number of users n yields higher uncertainty on the side of an attacker with respect to a specific user, and therefore improves the privacy level for most users—but it does not affect the amount of knowledge gained in the worst case (cf. Figure 17b).

4.5 PRIVACY-AWARE USER COUNTING

Even though naive user counting as discussed before is a promising approach which already protects the privacy of most users, we saw that in the worst case it is still possible to reveal the presence of some users. Now, the challenge is to leave an attacker deep in darkness also in extreme cases, while still getting accurate results. This aim in mind, we propose a perturbation technique.

With this technique, a service entry point will proceed just as discussed above and enable bits according to the hash coordinates of the locally observed user IDs. In addition, though, each individual bit in the sketch matrix will be set to one with a fixed, configured probability r.

By adding these randomly switched on bits, we add an additional source of "vagueness": the attacker does not know which bits have been randomly enabled. He therefore cannot make a definite decision whether a set bit has been set by a corresponding user or whether it has been switched on at random. The fact that the randomly enabled bits are uniformly distributed has two important implications. First, the bits on the right hand side in the sketch are now enabled with non-negligible probability, where, as discussed above, an attacker is otherwise able to gain very substantial knowledge. Second, when it comes to extracting estimates for the number of distinct users, the uniform distribution of the randomly set bits will allow us to separate their effects from the geometrically distributed bits introduced by "real" users.

We now analyze the impact of this modification on the worst-case user privacy. Subsequently, we discuss how accurate estimates can still be obtained.

4.5.1 *Privacy Analysis with Perturbation*

Adjusting our previous arguments, the probability that bit (i, j) is set if n users have contacted the service and, in addition, each bit has independently been set to one with probability r is

$$P((i,j) \mid n, r) = 1 - \left(1 - \frac{1}{m \cdot 2^j}\right)^n \cdot (1 - P(u)) \cdot (1 - r).$$

If we recalculate the a-posteriori probability $P(u \mid (i,j), r)$, now taking into account r, we arrive at

$$P(u \mid (i,j), r) = \frac{P(u)}{1 - \left(1 - \frac{1}{m \cdot 2^j}\right)^n \cdot (1 - P(u)) \cdot (1 - r)}. \tag{7}$$

Note that in case of $r = 0$, (7) is identical to (6).

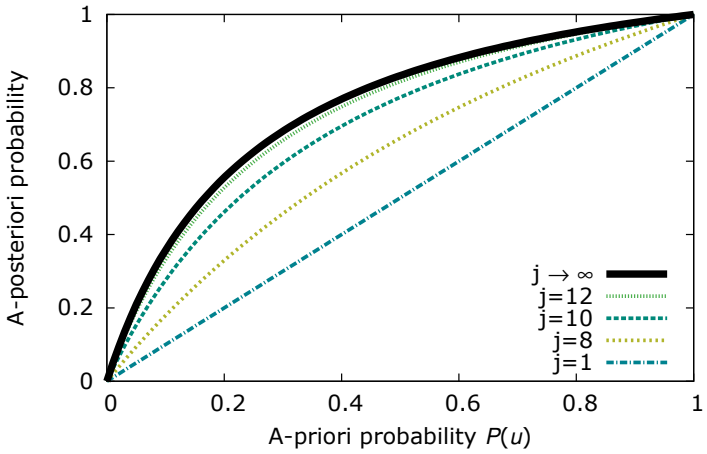


Figure 18: Knowledge gain with perturbation ($r = 0.2$, $m = 8$, $n = 1000$) – mitigation of the worst case: the a-posteriori probability no longer equals one; in general, r determines the privacy level.

The limit $j \to \infty$ of probability $P(u \mid (i,j),r)$, representing again the worst case, is

$$\lim_{j\to\infty} P(u \mid (i,j),r) = \frac{P(u)}{P(u) + r - r \cdot P(u)}.$$

Observe that the a-posteriori probability does no longer converge to 1 for $r > 0$; there is a significant remaining uncertainty for an attacker even for large j. We therefore succeeded to mitigate the worst case: regardless of the a-priori knowledge, the attacker can never be entirely sure.

Figure 18 depicts the improvement for $r = 0.2$ and the same values for m and n as before. The limit (8) is again drawn as a bold line. Note that the choice of r determines the worst-case privacy level: larger r means less information gain for an attacker.

To conclude, again consider the case where the attacker knows that there is only one single user ID mapping to a specific bit, and this bit is set to one. As pointed out before, this corresponds to the case $j \to \infty$, where the probability that other user IDs map to the same sketch bit approaches zero. For the naive approach, we found that the attacker will then inevitably gain definite knowledge. This is substantially different if perturbation is used, as for $r > 0$ the respective a-posteriori probability is no longer equal to one.

4.5.2 *Analysis of the Attacker's Information Gain*

A different perspective on the level of privacy with perturbed sketches can be obtained by explicitly calculating the information gain that an attacker gets by looking at the sketch. This can be quantified based on the entropy of the distribution of the attacker's knowl-

edge. To this end, we consider an attacker with minimum a-priori information about the presence of user u (i. e., an a-priori distribution with maximum entropy, equivalent to $P(u) = 0.5$). We then calculate the information gain of the attacker, which is given by the reduction of the entropy of the distribution of the attacker's a-posteriori knowledge compared to the a-priori knowledge. In this spirit, the attacker's expected information gain for a random user ID is given by

$$E[\Delta H] = \sum_{j=1}^{\infty} 2^{-j}(1 + p_j \log_2 p_j + (1 - p_j)\log_2(1 - p_j)),$$

where p_j is the a-posteriori probability (for $P(u) = 0.5$) for a user ID mapping to sketch column j. This probability is given by $P(u \mid (i, j), r)$ as derived in (7) above.

Evaluating this expression for, e. g., $m = 8$, $n = 1000$, and $r = 0.2$, it can be seen that the expected information gain of an attacker is as low as 0.00135 bits. And also for those IDs mapped to high bits in the sketch the picture remains favorable: even in the worst case of $j \to \infty$ and $r = 0.2$ an attacker with minimum a-priori knowledge cannot gain more than 0.34998 bits of information.

4.5.3 *Obtaining Estimates*

We have so far seen that the proposed perturbation effectively limits the attacker's knowledge gain. Now, the question arises how to still calculate accurate results despite the randomly added bits. Using the standard FM sketch evaluation formula as given above would lead to massive estimation errors, since the additional bits may increase the length of the initial sequence of ones in a sketch.

In [129], a related problem occurred. There, bits in an FM sketch randomly flipped to value one due to collisions of bit storage positions in the specific memory layout, which was designed for an application in the high speed networking context. In [129], this was not an intended feature, but rather an inevitable side effect. But even though the reasons for the perturbations and the problem setting are very different, the problem can be tackled with similar means. In short, the constant φ in Flajolet and Martin's estimation formula can be adapted according to the probability r. In analogy to Section 4.3, let $P(z \mid n, r)$ denote the probability that at least the first z bits are set after n additions. Taking r into account, we obtain a modified version of (3):

$$P(z \mid n, r) = \prod_{j=1}^{z} \left[1 - \left(1 - 2^{-j}\right)^n \cdot (1 - r)\right].$$

This formula can be understood as follows: a bit is zero if and only if it has neither been enabled by hashing a user ID nor by being

switched on randomly. The probability for *exactly* z initial ones is $P(z \mid n, r) - P(z+1 \mid n, r)$. This allows to obtain the expected value of the length of initial, uninterrupted sequence of ones Z as follows

$$E[Z \mid n, r] \quad = \quad \sum_{z=1}^{w} z \cdot (P(z \mid n, r) - P(z+1 \mid n, r)).$$

For $\varphi(n, r)$, now also depending on r, we get

$$\varphi(n, r) = \frac{2^{E[Z|n,r]}}{n}.$$

As in the case of unmodified FM sketches, $\varphi(n, r)$ converges very quickly to a constant value for large n; the limit, however, now depends on r:

$$\varphi(r) = \lim_{n\to\infty} \varphi(n, r).$$

Substituting φ in (5) by $\varphi(r)$ allows to obtain unbiased estimates even in the presence of perturbation.

4.6 COMBINING PERTURBED SKETCHES

One of our key aims is *distributed* counting of distinct user IDs. Therefore, we should consider the impact of combining multiple sketches. Assume that there are K service entry points (and thus K sketches) in total. Let $r_1, \ldots, r_K$ be the respective probabilities at which bits have been flipped to one. If these sketches are combined with a bit-wise logical `OR` in order to estimate the total number of distinct users, the result is equivalent to a single sketch for all users which has been perturbed at probability

$$r^* = 1 - \prod_{k=1}^{K} (1 - r_k). \tag{8}$$

The combined sketch can thus be evaluated by applying the above methodology, using $\varphi(r^*)$ in the evaluation formula.

It would be easy to collect all sketches at one central point and to combine them afterwards. As long as each sketch is perturbed with a sufficiently high probability, the collector cannot gain any significant knowledge. However, combining many perturbed sketches quickly drives r^* to exceedingly high values. This would result in bad accuracy if sketches from a larger number of service entry points are combined. On the other hand, if sketches are perturbed with only a low probability (in order to keep r^* low), the collector could gather substantial knowledge by looking at each sketch.

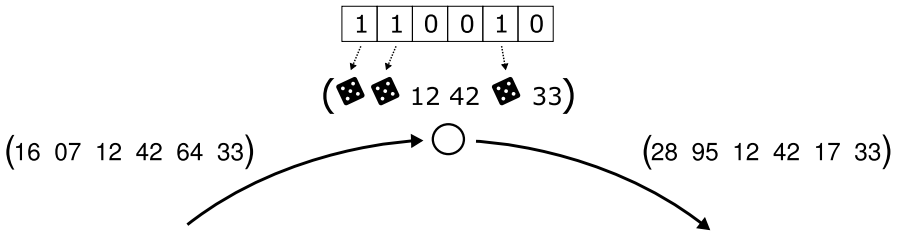


Figure 19: Combining sketches in a privacy-preserving way without leaking any information in the presence of an honest-but-curious adversary.

This raises the question whether sketches can be collected and merged in such a way that no instance in the system is able to see the individual sketches of any other service entry point. In the following, we show how this can be accomplished, in line with the honest-but-curious adversary model. The benefit of this technique is that relatively low perturbation probabilities can be used for the individual service entry points' sketches, so that in total for the merged sketch a reasonable tradeoff between privacy and estimation accuracy is achieved.

We arrange all service entry points $N_1, \ldots, N_K$ in a logical circle. One of the entry points, w.l.o.g. N_0, is designated to start the collection process. N_0 then generates a matrix M_0 filled with random numbers. This matrix has the same dimension $m \times w$ as the used sketches. M_0 is temporarily stored by N_0, and it is transmitted to the next service entry point along the circle, N_1. M_0 is not revealed to any other entity.

Upon reception of the matrix, N_1 generates a new matrix M_1. At those matrix positions where N_1's local perturbed sketch has a bit with value zero, the entries in M_1 are set equal to the corresponding entries in M_0. Where a bit in N_1's local sketch is one, a new random number is chosen for the respective entry in M_1. This new matrix M_1 is subsequently passed on further along the circle to N_2, where the same procedure is executed to generate a matrix M_2, and so on. This procedure is illustrated in Figure 19.

After the last service entry point N_K has generated matrix M_K, this matrix is passed on to N_0.

N_0 can then determine the combined sketch by comparing M_0 and M_k. Each sketch position that has been set by at least one of the service entry points will have been set to a new random value at least once while the matrix is passed around. Each position that has not been set at any service entry point will remain at the original value it had in M_0. When receiving M_k, N_0 thus considers a position as a set bit in the merged sketch if and only if it differs from the corresponding entry in M_0. From the thereby obtained sketch matrix, the number of distinct users can be derived as discussed above.

Of course, it is conceivable that a matrix position is "accidentally" set to the very same random value that it had in M_0 by the last entry point modifying this position. This possibility could be avoided by incrementing the respective random entry instead of drawing new random numbers when a sketch entry is set (essentially "adding" all sketch matrices to the original random matrix M_0). This, however, would reveal the number of entry points at which a given bit is set to N_0. Therefore, it seems wiser to set the entries to new, independent random values: after all, if the value range for the matrix entries is large enough (e. g., 32 bit numbers), the probability of accidentally restoring the original random number can safely be neglected for any practical purposes.

This procedure ensures that no instance in the system, in particular neither intermediate entry points nor N_0, can see any individual sketch matrices except their own one. Only the combined matrix from all service entry points is revealed to N_0. Thus we can apply a relatively small perturbation probability r to each individual sketch, without the risk of internal or external attackers gaining too much knowledge.

The random numbers in the passed-on matrices are of no value unless other, earlier matrices are known. In case of colluding adversaries, who surround a specific single entry point in the logical circle, they can obtain the local sketch of the surrounded node by comparing their sketches before and after it was passed-on. The more entry points lie between the adversaries, the higher the combined perturbation probability. Hence we must ensure that the sequence of nodes in the logical circle is randomized, so that it becomes unlikely that colluding adversaries are grouped together and that they cannot intentionally surround a specific node.

The remaining question is how the probability r should be chosen. Reasonably, the same perturbation probability is chosen for all service entry points, i. e., $r_1 = r_2 = \ldots = r_K = r$. (8) thus becomes

$$r^* = 1 - (1 - r)^K,$$

and solving for r yields

$$r = 1 - (1 - r^*)^{1/K}.$$

For this choice of r, the desired perturbation probability r^* of the combined sketch is achieved. For example, for a reasonable target perturbation probability of $r^* = 0.1$ and for $K = 1600$ service entry points, each entry point should perturb its sketch with a probability of $r \approx 6.6 \cdot 10^{-5}$. Please note again that, such small perturbation probabilities are feasible, because the developed combination algorithm does not reveal individual entry points' sketch matrices to any other system entity.

4.7 MULTIPLE OBSERVATION INTERVALS

A somewhat subtle issue arises in a setting where sketches are generated not once in a single observation interval, but multiple times—e. g., if the number of users is monitored continuously on a daily (or hourly,...) basis. This is problematic if a user's ID maps to a bit in a far-right column (high j), and an attacker is able to monitor the sketches over multiple intervals: if the respective bit is set to one in many observation intervals, then the attacker may conclude with high probability that the user regularly uses the service—simply because it is unlikely that the respective bit has *regularly* been randomly set to one. In order to overcome these difficulties, we propose to change the hash functions h_1, h_2 between observation intervals, by including a hash salt. All service entry points need to agree on the same salt for the same observation interval. The used hash salts need not be kept secret. For instance, the observation intervals' starting point might be used as salt.

With this modification, the mapping of an ID to a position in the sketch will vary from one to the next observation interval. In general, any ID will map to one of the "uncritical" bits with low j for the vast majority of intervals. The very low expected attacker information gain for a random user ID derived above readily shows that even for huge numbers of observation intervals it will not be possible for an attacker to accumulate significant knowledge.

4.8 EVALUATION

In the preceding sections, a detailed analysis underlined the privacy-preserving nature of our approach. Now we shift our focus to the accuracy aspect. To this end, we perform evaluations in a simulated setting, where the exact number of simulated users is known without the danger of harming real users' privacy. We will assess the accuracy impact of the parameters in our algorithm, and we compare it to the accuracy obtained with user count estimation based on data from a single entry point, which resembles Tor's approach.

4.8.1 *Impact of Perturbation on the Accuracy*

Let us first assess the impact of the perturbation probability r on the estimation accuracy. As one might expect, increasing r deteriorates the accuracy to a certain extent. In order to quantify this effect, we simulated our proposed algorithm by generating sketches into which we inserted 1000 randomly generated distinct user IDs each. We then additionally switched bits to one with varying probability r, and subsequently extracted estimates from the sketches. In Figure 20, we show the standard error of the estimates in relation to r for differ-

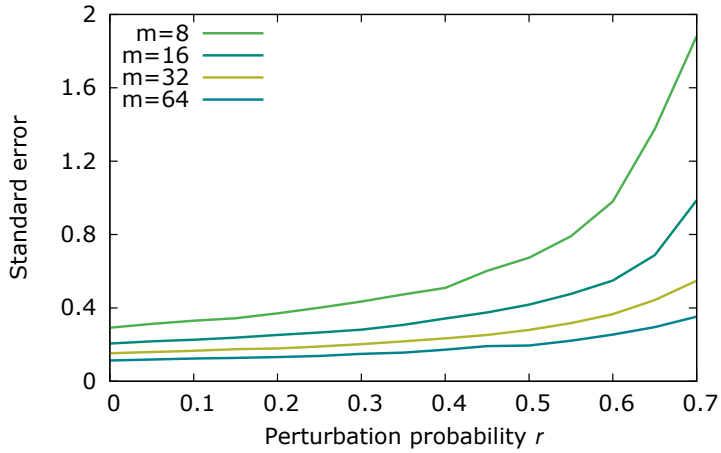


Figure 20: Standard error of the estimation: for increasing r, the standard error increases, but remains especially for higher m at reasonable levels.

ent values of m; we used 8000 independent samples per data point. As for original FM sketches, increasing m reduces the standard error. For increasing r, the standard error increases, but remains at reasonable levels even for relatively high values of r. This holds especially for higher sketch row counts m.

4.8.2 *A Comparison to Request-Based User-Counting*

We will now present simulation results from a system with 1,600 entry points and 100,000 simulated users connecting to the system. Each user makes a random, Poisson distributed number of requests to randomly chosen entry points.

In this setting, we compare our algorithm—which is able to collect and merge information from all entry points—to estimating the total user count from information on the number of users that connected to one single entry point. As we will demonstrate (and as pointed out before, e. g., in [96]), the latter approach is potentially highly inaccurate. The key reason is that it is necessarily based on non-verifiable assumptions about the distribution of user requests. In order to see why, consider the following two extreme cases: in the first case, each user makes exactly one request, to a single entry point chosen at random. Assume, for instance, that we expect to see 1 % of the total users at one specific observed entry point. Then, the locally observed user count should be multiplied by a factor of 100 in order to obtain an estimate for the total number of users. In the other extreme case, each user contacts all the entry points. In that case, multiplying the number of users seen at one entry point by a factor of 100 would lead to gross overestimation. One cannot tell where in between these extremes a system currently operates without exchanging information

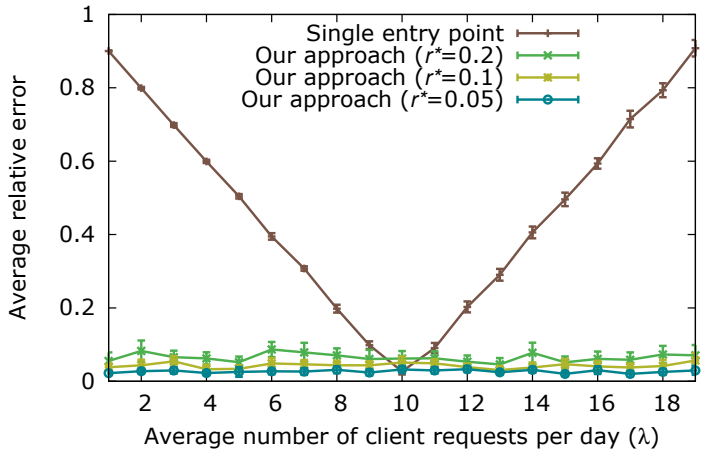


Figure 21: Simulation results (25 runs, 1,600 entry points, 100,000 users, $m = 64$): the accuracy of our approach is independent from the average number λ of user requests per day.

about individual users between entry points (and, after all, avoiding the need to exchange such data is the key aim of this work).

Tor, for instance, currently estimates the number of users based on the requests seen by directory mirrors [59, 132, 134]. To this end, it is necessary to estimate the share of directory requests that select a mirror, which is done based on the directory mirror selection algorithm in Tor and on the currently active set of directory mirrors. Tor's current user count estimation mechanism then assumes that an average client performs ten requests per day. This is a guess at best, because there is no data available to substantiate this assumption. It is thus a source of potentially very significant inaccuracies.

Our algorithm, in contrast, is able to merge data from multiple entry points without sacrificing duplicate insensitivity. It does therefore not need to estimate the total user count based on a limited local view. Consequently, it eliminates the above mentioned sources of inaccuracy due to unknown user behavior. The price that is to be paid in exchange is the inherent, but limited and well understood inaccuracy of perturbed FM sketches.

These properties are illustrated by our simulation results. They are based on the above outlined setting, and are shown in Figure 21. All results are averages from 25 independent simulation runs, the error bars show 95 % confidence intervals. On the x axis, we used different values of λ for the Poisson distributed number of requests per user. A value of $\lambda = 10$, for instance, means that an average simulated user makes ten requests. The y axis shows the resulting average relative estimation error.

The figure shows results obtained with our algorithm for different values of the perturbation probability r. As already observed in the previous subsection, the accuracy of course depends on r: more perturbation results in somewhat higher errors. However, here it also

becomes clear that the accuracy is independent from λ: the estimates are obtained from duplicate-insensitive, combined information from all entry points. They do thus not depend on how many requests each user makes.

The number of sketch rows, m, has always been set to 64 here. As discussed above, increasing m results in more accurate estimates.

We compare these results with a user count estimation based on data from a single entry point. Like those in Tor, these estimates are based on the assumption that each user makes, on average, ten requests per day. It can clearly be seen that this results in very accurate estimates if and only if this (in practical applications non-verifiable) assumption holds: if $\lambda \approx 10$, the results are very good; otherwise, they can be arbitrarily far off. The credibility of statistics that are based on non-verifiable assumptions about user behavior is thus generally questionable.

Consequently, the results underline the key benefit of our approach: it allows to obtain global, privacy-preserving, and duplicate-insensitive user statistics without the need to make assumptions about user behavior that cannot be checked.

4.9 SCOPE AND APPLICABILITY IN OTHER AREAS

Counting the number of users in an anonymity network like Tor serves as a prime example for the practical applicability of our approach. Beyond this specific scenario, there are plenty of other applications where it can be useful.

Distributed databases could use our algorithm to query for aggregated statistics. For example in the context of health care, one could ask for the number of patients with a certain disease. Here, all entities (i. e., entry points), which could be doctors' offices or hospitals, agree on the same unique identifier for each patient (e. g., the social security number). According to our algorithm, each entity populates its local sketch with all patients suffering from this certain disease. After passing around and combining the perturbed sketches, the resulting statistic can be revealed without being able to identify any patient individually. Since our algorithm has the property of being duplicate-insensitive, patients can be under treatment at multiple places without producing an overestimation.

In order to collect various attributes at once or a distribution of data, all entry points hold multiple sketches in parallel, each representing an attribute. For instance, to count the number of users by country, each country has a designated sketch. Due to the relatively small size of the exchanged sketches and the low bandwidth footprint of our scheme, even statistics per data element or per search term appear feasible: with separate sketches for each from a set of interesting

data items or search terms, the number of distinct users accessing the respective elements could be determined.

Finally, it should be noted that the data structures introduced here cannot only determine the number of distinct elements, but also provide a basis for other statistics. For instance, sums can be calculated by generating/adding a respective number of "dummy" elements (these elements need not be added individually, but a corresponding set of bits to be enabled in the sketch can be generated more efficiently). Products can be expressed as a sum of logarithms. Based on two sketches, duplicate-insensitive distributed averages can be calculated: one for a sum of values, another one for the number of distinct entries. Sketches inherently support union operations. Intersections can be realized for instance by using the principle of inclusion and exclusion. All these (and many more) operations can be reduced to adding elements to a sketch. Therefore, they can all be implemented in a distributed, privacy-preserving way based on the algorithms introduced here.

4.10 RELATED WORK

Tcpdpriv [144] is a tool for anonymizing network trace files, and thus faces related challenges. It maps IP addresses to anonymized representations while maintaining prefix congruence. An advanced alternative is [79]. To determine the number of distinct IP addresses across multiple service operators, preserving prefix congruence does not suffice.

In the database context, privacy-aware data aggregation and summarization techniques have also been considered [175, 198]. To answer statistical questions without compromising privacy, perturbation techniques are common, as they are discussed for example in [16, 74, 145, 163, 172]. Similar to such approaches, we add randomness in order to obscure data. However, all these techniques do not allow to obtain the statistics we are interested in: the number of distinct data elements across multiple sources.

There exist several approaches which provide private set operations [47, 58, 87, 99, 138]. They are based on cryptographic means such as secure multiparty computation (MPC). These approaches share a number of shortcomings including high computational costs, huge communication overhead and/or a very limited maximum number of feasible participants.

SEPIA [47], a secure MPC framework, for example, provides a distributed algorithm for privacy-preserving, distinct counting. It is based on Shamir's secret sharing [182]. In this algorithm, lists of items are shared by so-called input peers among a set of privacy peers. Input peers, which correspond to our entry points, record data. Privacy peers perform calculations using secure MPC and finally reveal the

result. The key problem with such an approach in applications like ours is the limited scalability. Multiple distributed per-element computations with intermediate data exchanges are necessary. As stated in [47], a 16-bit ID space with 25 input peers and 9 privacy peers results in 50 MB of communication overhead per peer. This increases linearly with the size of the ID space and the number of peers. There are 2^{32} IPv4 addresses, so for counting distinct user IP addresses in the IPv4 Internet the resulting overhead already exceeds all reasonable bounds and becomes intractable. IPv6 would make things even worse.

Bloom filters [39] can be applied as privacy enhancing technique to the private matching problem [124, 177] or to enable private database queries [36] for example. They can also be used to estimate the distinct number of elements in a set [158]. However, the use of Bloom filters exhibits severe drawbacks for our objective. In particular, the Bloom filter needs to be appropriately dimensioned beforehand. This requires approximate knowledge of the number of expected elements, because the required Bloom filter size for a given accuracy and a given number of used hash functions increases linearly with the element count. If the Bloom filter's bit field is chosen too small for the number of occurring elements, the accuracy rapidly deteriorates. A too large Bloom filter, on the other hand, results in a very low false positive rate; individual entries can then be identified with high probability, which thwarts privacy.

In [138], the developers of SEPIA recently proposed to apply Bloom filters [40] to their framework to estimate the distinct count for larger ID spaces. By doing so they eliminate the linear growth of the communication overhead with the ID space. However, the problem of dimensioning the Bloom filter appropriately also exists in this setting. Moreover, also with Bloom filters the total bandwidth demand still increases quadratically with a growing number of participants. For our approach the communication overhead per participant is constant.

Our proposed algorithm is based on an algorithmic foundation of FM sketches [81]. FM sketches have been used before in other contexts [57, 130, 190] where duplicate insensitive multiset cardinality estimates are needed. Neither the original contribution nor any of these applications take the protection of privacy into account, though.

In our privacy analysis we use the notions of a-priori and a-posteriori knowledge for an adversary's information before and after looking at the exchanged information. These terms are often used and well-known, but only loosely defined [162]. Here, we identify the attacker's a-priori and a-posteriori knowledge with the a-priori and a-posteriori probabilities of a specific element or user being present. The latter terms are much more clearly and precisely defined, and therefore provide a handhold for rigorous formal analysis. Where a-priori and a-posteriori knowledge have been used in previous work

in a spirit similar to ours, it was either not possible to express non-definite, probabilistic knowledge [53, 140], or zero a-priori information is explicitly assumed [65, 178]. In fact, these works explicitly point out that the key challenge in analyzing the degree of privacy is to appropriately incorporate the adversary's a-priori knowledge. In this chapter, we deliver a formal analysis which quantifies these interdependencies for arbitrary a-priori probabilities, including a worst-case bound for an attacker's knowledge gain.

4.11 CHAPTER SUMMARY

We presented a methodology to count users based on their observed user IDs in a distributed and privacy-preserving manner. The algorithmic properties of FM sketches provide a way to deal with duplicate occurrences of user IDs. An evaluation of their naive application, however, revealed severe shortcomings with respect to user privacy, at least in the worst case. We showed that this can be overcome by a perturbation technique that mitigates the worst case so that *all* users' privacy is protected.

In our analysis, we compared an attacker's a-priori and a-posteriori knowledge and the corresponding entropy difference in order to quantify the information gain. This perspective allowed us to discuss parameter impacts and tradeoffs also in those situations where the attacker cannot become entirely sure about the presence of a user. We showed that our proposed algorithm effectively limits the possible information gain of an attacker. It therefore guarantees that an attacker can never be sure that a specific user has used the service.

In order to still calculate good estimates from our modified FM sketches, we showed how the evaluation methodology needs to be adjusted. Furthermore, we identified several parameters that help adjusting the tradeoff between accuracy and privacy.

We believe that our algorithm can provide valuable data to assess the "health" of the Tor network and support future design decisions. The applicability, though, reaches well beyond estimating the number of users: it can also be used in other situations where one is faced with the problem of distributed, privacy-aware, duplicate-insensitive counting.

5

THE SNIPER ATTACK: DISABLING THE TOR NETWORK

5.1 OVERVIEW

This chapter is based on previous collaborative work [1]. In particular, conducting the evaluation should mostly be attributed to fellow co-authors.

Since Tor is the most popular deployed system for fighting censorship and online privacy encroachments, it also poses an attractive adversarial target. We argue that understanding possible attack vectors is paramount not only to the successful design of future technologies, but also to the security of existing networks and systems.

In this chapter, we present an extremely low cost but highly destructive denial of service (DoS) attack against Tor relays. By simply disabling all relays or intelligently targeting crucial subsets of relays, our attack can be used to disable the entire Tor network. In addition to threatening network availability, it can be used to facilitate hidden service deanonymization. Our attack thus imposes real, significant threats to Tor users and we believe it constitutes the most devastating attack against the Tor network to date.

We disclosed our attack to the Tor Project and have worked together to deploy a defense [66, 141]. As a result, Tor is no longer vulnerable since version 0.2.4.14-alpha.

The attack, which we call the *Sniper Attack*, because the attacker remains hidden while disabling relays in a targeted manner, works by exploiting Tor's reliable data transport and flow control mechanisms. In particular, an adversarial client builds a regular Tor circuit using the target relay as the entry, commands the exit to start downloading a large file through the circuit, *without reading* from the target entry. By circumventing Tor's flow control, the exit will continue to pull data from the external data source and push it into the circuit. The target relay is forced to buffer the data in application queues. Eventually, memory becomes exhausted, resulting in termination of the Tor process by the operating system's memory manager (e. g., the oom-killer on Linux [155]). The evaluation of our attack prototype in a safe, private, simulated Tor network demonstrates the destructiveness and confirms its low cost.

The results underline our arguments provided in Chapter 3, i. e., intertwined mechanisms of underlay and overlay transport cause undesired interferences. In particular, the additional instance of data forwarding and transport functionality on top of the existing Internet protocols is the main source of interferences in Tor.

Exploring defense strategies against the Sniper Attack clearly illustrates the difficile situation. We discuss how simple hard-coded queue size limits and end-to-end authenticated signals affect the adversary's attack strategy, but are not able to completely prevent the attack. We then present an algorithm that adaptively reacts to high memory pres-

sure indicative of the attack. Our adaptive defense utilizes queuing delay as a heuristic to identify and kill malicious circuits in order to protect the Tor process from being killed. We derive resource bounds with our defense mechanism in place, showing that it cannot reasonably be leveraged by attackers to cause relays to destroy benign circuits. Our evaluation shows that our adaptive circuit killing defense detects and stops the Sniper Attack without any false positives.

Our contributions can be summarized as (a) designing a dangerous and destructive DoS attack capable of disabling arbitrary Tor relays and (b) developing practical defenses against the Sniper Attack that reduce Tor's vulnerability to attacks that target Tor's queuing mechanisms. In general though, the key insight is the identification of a fundamental design issue, which reaches far beyond anonymity networks: performing hop-by-hop reliability and end-to-end flow control is an unfavorable design decision, which inevitably implies vulnerabilities and other disadvantages.

5.2 THE SNIPER ATTACK

In this section, we develop a DoS attack against the Tor network that can be used to anonymously disable arbitrary Tor relays by killing the Tor process on its host machine. To facilitate an understanding of the exploited protocol features, we first describe a basic attack that requires the adversary to run both a Tor client and a Tor exit relay. We then describe a more efficient variant that only requires a Tor client and therefore significantly reduces the resources required by the adversary. Finally, we discuss strategies that disguise the adversary's identity.

5.2.1 *Basic Attack*

Recall that Tor is an overlay network and clients are responsible for path selection. Each pair of relays communicate over a single TCP connection. The application layer protocols rely on this underlying TCP connection to guarantee reliability and in-order delivery of application data between each relay. As a result of using hop-by-hop TCP at the network layer, Tor does not allow relays to drop or re-order cells at the application layer.

The Sniper Attack exploits Tor's reliable application-level queuing. Our assertion is that if a relay *stops reading* from a connection it will cause the respective downstream hop to buffer a full package window worth of data (1000 cells) for every active circuit multiplexed over the connection. This requires at least two streams per circuit and that streams in sum transfer enough data to drain a circuit-level package window. When a relay with incoming data stops reading from its TCP socket on the connection to an adjacent relay, the TCP receive

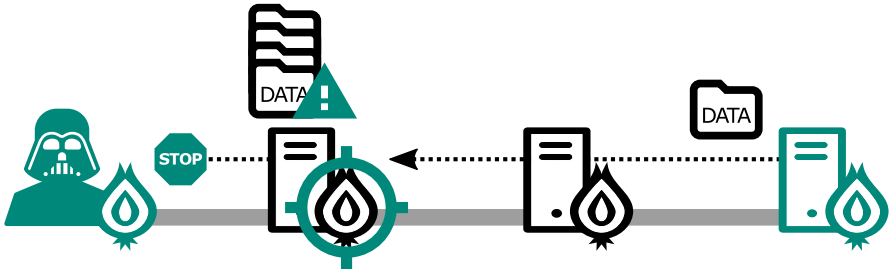


Figure 22: Basic version of the Sniper Attack.

buffer will fill, TCP flow control window will empty, and therefore announce a zero window to the other end of the TCP connection. The adjacent relay will then no longer be able to forward cells, causing its TCP send buffer to fill. With a full TCP send buffer, the adjacent relay will buffer cells in the application layer circuit queue (again note that Tor does not allow relays to drop cells in the application layer) until the stream or circuit package window reaches zero. At this moment edge relays will stop injecting new data into the circuit.

Tor provides parameters, `EntryNodes` and `ExitNodes`, to specify a list of relays for the respective roles. Likewise the Tor control protocol [193] allows to build custom circuits.

Using the mechanism described above, an adversary that controls a client and a relay may attack a target relay as shown in Figure 22. The adversarial client constructs a circuit by selecting the target relay as the entry and the adversarial relay as the exit. The client signals the exit to start the attack by issuing an arbitrary request over the custom attack circuit, and then stops reading from the TCP connection to the target entry. The exit simply ignores the empty package windows and continuously sends data it arbitrarily generates, increasing the amount of memory consumed by the entry to queue the cells. Note that it is not necessary for the malicious exit to produce correctly encrypted Tor cells since they will never be fully decrypted by the client (though correct circuit IDs are required). Eventually, the Tor process on the entry node depletes all of the available memory resources and is terminated by the operating system. On Linux systems, this job is handled by the out-of-memory (oom) killer [155].

Note that the adversary may choose any relay as its target entry. However, choosing relays without the guard flag for a circuit's entry position will raise suspicion since Tor's default path selection algorithm will not choose entries in that manner. Therefore, the attack may be slightly modified to target any middle or exit node as well.

The TCP connection from the client to the target must remain open from the victim's perspective to prevent the attack circuit from being closed and its queue cleared, but the cost of doing so is insignificant (and it can be done without maintaining state [105]). Also, the adversary may slightly reduce the required bandwidth by minimizing the size of its TCP receive buffer, e.g., by using `setsockopt` [191].

5.2.2 *Efficient Attack*

We now describe an efficient Sniper Attack that eliminates the necessity of generating and uploading data, thereby significantly reducing resource demands. This efficient version of the Sniper Attack exploits Tor's end-to-end flow control signals. Our assertion is that the `SENDME` cells only *imply* that data has been successfully received and edge relays may send `SENDME`s without actually receiving any data.

The efficient Sniper Attack works by combining the `SENDME` signal mechanism described above with the stop reading mechanism from the basic version of the attack. As shown in Figure 23, the adversary must only control a single malicious client. This client first builds a custom circuit by selecting the target as the circuit entry, and then initiates the download of two large files (e.g., large Linux distributions) over the circuit to ensure that the two streams will empty the exit's circuit package window. The client then stops reading from the connection to the target entry, and begins maliciously sending `SENDME`s to the exit to ensure that the exit's package window does not reach zero and it continues injecting packaged data into the circuit. These packaged cells will continue to flow to and be buffered by the entry in its application queue, continuously consuming memory until the entry's Tor process is selected and killed by the OS.

AVOIDING DETECTION To launch a successful Sniper Attack, the adversary must circumvent a protective mechanism that Tor employs to prevent protocol violations, e.g., by clients who try to cheat by sending more `SENDME` cells to get more data earlier. When the exit relay receives a `SENDME` that causes its circuit window to go above 1000 cells, it detects the violation, closes the circuit, and sends a `DESTROY` cell backwards. The middle hop converts the link-level `DESTROY` cell into a `RELAY` cell of type truncate and sends it to the entry, who just passes it back to the client. When the client extracts the `DESTROY` cell (that originated at the exit) from the `RELAY` cell, it closes the circuit and sends a `DESTROY` cell forward to the entry. The entry closes the circuit (clearing the circuit queue) and forwards the `DESTROY` cell to the middle, who also closes the circuit.

In order for the attack to succeed, the adversary ideally would (a) prevent the exit's package window size from exceeding its size; and (b) in case it does, the client would avoid sending out the final `DESTROY` cell to ensure the entry does not clear its queue. Note that since the malicious client will not be reading from the target entry, the adversary will not be able to determine if (a) occurred, and therefore does not need to handle (b) in practice. However, we note it here for completeness. Also note that, as will be discussed in the next section, even if the adversary fails at (a) and the exit detects a protocol viola-

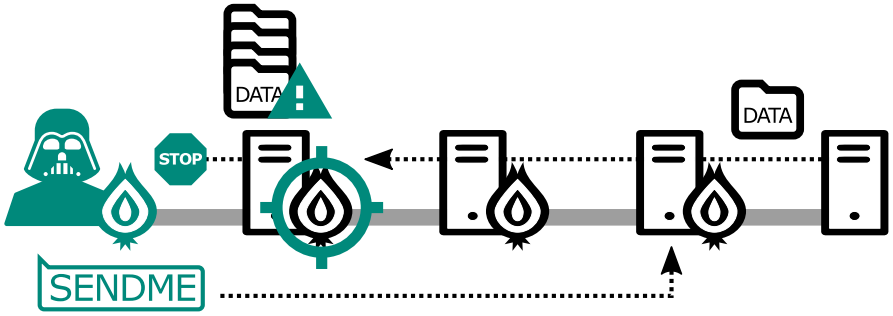


Figure 23: Efficient version of the Sniper Attack.

tion, the attack circuit will continue to consume the target's memory until the TCP connection is destroyed.

The adversary may avoid the exit's protective mechanism by sending SENDMEs to the exit at a rate low enough so that the exit's package window never exceeds 1000 cells. One approach to estimating such a rate is to consult the Tor Metrics [195] and use recent relay byte histories to estimate the throughput of the custom circuit. However, given the dynamics of the Tor network and its usage, this approach would likely result in a high failure rate. Instead, a malicious client may account for real time congestion by performing file download probes through the same nodes that were chosen for the target circuit. If each probe downloads σ kB in Δ seconds, then we can estimate the circuit throughput as σ/Δ kiB/s, or $2\sigma/\Delta$ cells/s (all Tor cells are 512 bytes in size). Now recall that stream and circuit level SENDMEs are sent for each 50 and 100 downloaded cells, respectively. Thus, using our probe we estimate that stream and circuit level SENDMEs be sent every $T_{stream} = 25\Delta/\sigma$ seconds and $T_{circuit} = 50\Delta/\sigma$ seconds, respectively. The malicious client may update Δ by periodically performing an additional probe, and larger values of σ are more costly but will produce more accurate estimates over time. Probing requires additional adversarial bandwidth, but this cost may be significantly reduced.

PARALLELIZING THE ATTACK Recall that the exit will close a circuit if the package window exceeds its size, and this circuit closure will be undetectable by the client once it stops reading from the target entry. Although a circuit closed by the exit will not cause the target entry to clear its application queue (and therefore free any memory consumed by that circuit), the circuit may no longer be utilized to increase memory consumed by the target entry. This situation may occur even if the adversary probes the circuit to find a good SENDME rate, since relay congestion and path throughput are highly dynamic.

To improve the attack's resilience to circuit closures while at the same time speeding up the rate at which the target's memory is consumed, the adversary may parallelize the attack by using multiple teams of multiple circuits. One circuit in each team is assigned prob-

ing duties (in order to measure Δ as described in the previous section), while the remaining circuits are assigned `SENDME` sending duties (to cause the exit to push data toward the target). The Δ computed by a team's probing circuit is used to dynamically inform the rate at which that team's sending circuits send `SENDME`s. Each team is assigned a Tor path using the target as the entry relay and uses that path to build each of its circuits.

We now consider how these circuits are constructed. Recall that once the attack begins and the adversary has stopped reading from the onion-routing connection to the target, it will be unable to determine which circuits on that connection have closed and which ones have not, and will also be unable to create new circuits over that connection. Since a separate connection is required for the probing circuits (because it must communicate bi-directionally), the adversary will need at least two connections to the entry for each team if the attack is to be successful. With this in mind, we consider three viable attack strategies: (a) use one Tor client instance for each circuit of each team; (b) use one Tor client instance per team that creates a new onion-routing connection to the target whenever one is needed; and (c) use two Tor client instances per team: one that controls the probing circuit and one that controls the sending circuits. Note that unique onion-routing connections are guaranteed by using separate Tor client instances. Although each of the above strategies are viable, we reject (a) because there is a high resource cost associated with running many Tor instances, and we reject (b) because multiple connections from a single Tor client instance would be easy for the entry to detect and would require significant code changes. Therefore, we assume the adversary uses strategy (c) where all circuits are operating in parallel.

The use of multiple circuits within each team will increase the throughput achieved by that team from its assigned path due to the circuit scheduling policies employed at each relay and will prevent a single sending circuit failure from stalling the attack. Using a consistent path within each team ensures that the sending rate Δ is accurate for all of that team's members. Assigning middle and exit relays independently for each team further utilizes Tor's distributed resources by reducing the effect of throughput bottlenecks while also increasing the robustness to node failures. Finally, as there is no circuit feedback, the adversary may also pause the attack on existing teams and rotate to new ones over time to ensure that the target entry's memory consumption continues to increase.

5.2.3 *Hiding the Sniper*

For simplicity, we have thus far discussed the Sniper Attack as if the adversary is directly connecting to the target entry. Here, $\mathcal{C}$ denotes

client, $\mathcal{G}$ denotes entry, $\mathcal{M}$ denotes middle, $\mathcal{E}$ denotes exit, and $\mathcal{S}$ denotes server, while the subscripts A and V denote adversary and victim, respectively. The path of the attack as previously described may then be represented as:

$$\mathcal{C}_A \leftrightarrow \mathcal{G}_V \leftrightarrow \mathcal{M} \leftrightarrow \mathcal{E} \leftrightarrow \mathcal{S}$$

In this situation, the victim $\mathcal{G}_V$ knows the adversary $\mathcal{C}_A$'s IP address since they are directly connected. $\mathcal{G}_V$ may have enough information to blame $\mathcal{C}_A$, either during or after the attack, because of the anomalous behavior. Extra protections may be desired to avoid this exposure.

STEALTH WITH TOR Tor itself is a useful tool to provide such protections. One way the adversary could use Tor is by also running a Tor exit node:

$$\mathcal{E}_A\mathcal{C}_A \leftrightarrow \mathcal{G}_V \leftrightarrow \mathcal{M} \leftrightarrow \mathcal{E} \leftrightarrow \mathcal{S}$$

This situation provides the adversary *plausible deniability*: $\mathcal{G}_V$ will not be able to distinguish an attack by $\mathcal{C}_A$ from one launched through a circuit in which $\mathcal{E}_A$ is merely serving as the exit. However, drawbacks to this approach are that $\mathcal{E}_A$ will need to serve as an honest exit, which consumes far more resources than required by the attack and also results in the adversary appearing in the public Tor directory. The adversary then has to ensure that $\mathcal{E}_A$ has the characteristics of other honest exits (has the right consensus flags for its activities, has the right amount of traffic for its consensus weight, etc). Further, $\mathcal{G}_V$ will still know the IP address and may use it as a starting point when looking for someone to blame.

Alternatively, the adversary may use a full Tor circuit:

$$\mathcal{C}_A^2\mathcal{C}_A^1 \Leftrightarrow \mathcal{G}^1 \Leftrightarrow \mathcal{M}^1 \Leftrightarrow \mathcal{E}^1 \leftrightarrow \mathcal{G}_V^2 \leftrightarrow \mathcal{M}^2 \leftrightarrow \mathcal{E}^2 \leftrightarrow \mathcal{S}$$

This provides the adversary *anonymity*. It will prevent A's IP address from being known by anyone except $\mathcal{G}^1$, who will be oblivious to the attack. In this scenario, $\mathcal{C}_A^1$ stops reading on the connection to $\mathcal{G}^1$ but $\mathcal{C}_A^2$ sends `SENDME`s to $\mathcal{E}^2$ through the $\mathcal{C}_A^1$ proxy tunnel. A drawback to using a separate circuit in this way is that it may slightly increase the latency and length of the attack, because $\mathcal{G}_V^2$ will not start depleting its memory resources until $\mathcal{E}^1$'s package window reaches zero. It may also be more difficult to estimate a good `SENDME` rate when concatenating two circuits, and the adversary must now run twice as many Tor client instances to ensure that each team has two anonymous tunnels. Finally, a circuit that exits back into Tor may draw unwanted suspicion.

STEALTH WITHOUT TOR Alternatives to using Tor to hide include using public open wireless access points, briefly renting a small botnet, or using a cloud computing system. However, more entities will

then know about the adversary's actions, increasing the risk of discovery: access points and cloud services will be collecting logs; and some number of bots could be part of a honeypot. The adversary may want to connect to these services through Tor anyway to remain anonymous to them, and the composition of services will make it easier to make a mistake. By using Tor as described above, the adversary does not need knowledge of botnets or cloud systems, drastically simplifying the attack.

5.3 EVALUATION

We implemented a prototype of the Sniper Attack in order to evaluate its feasibility and efficacy. We evaluated it using Shadow [109], a discrete event network simulator that runs Tor code in a private Tor network, after testing its functionality in a minimal private Tor network in our lab. Shadow enables a safe development and evaluation environment that does not harm the security and privacy of the operational Tor network or its users, while still simulating application layer behavior in its full complexity since it runs authentic Tor code. In this section, we detail our private Tor network configuration, describe our prototype implementation, evaluate the attack's efficiency and resource costs, and analyze our results in the context of the live Tor network.

5.3.1 *Private Tor Network*

Tor nodes running in Shadow communicate over a simulated network. Therefore, Shadow requires models of downstream and upstream node bandwidths as well as link latency, jitter, and packet loss rates. The Shadow distribution includes these models, and also includes tools to generate private Tor network configurations for running Shadow simulations. Using these tools and archived Tor directory data (as of 2013/06/30) published by Tor Metrics [195], we configure a private Tor network consisting of 4 directory authorities, 400 relays, 500 file servers, and 2800 clients. This private network consumes roughly 64 GB of memory on our Linux host during each experiment. The clients generate background traffic during the experiments by downloading variously sized files from the servers through our private Tor, causing congestion and performance characteristics indicative of conditions in the live Tor network. All of these nodes run in the Shadow simulator and communicate only with one another. Our configuration follows the methodologies from [107], which describes in detail the modeling choices made by Shadow's configuration generation tool.

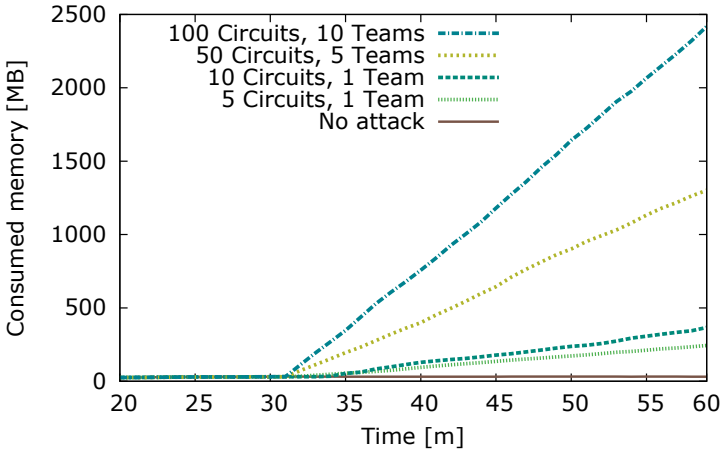
5.3.2 *The Sniper Attack Prototype*

We implemented the parallel version of the efficient Sniper Attack as described in Section 5.2.2, including multiple parallel circuits but without the rotating circuits enhancement. In our prototype implementation, a *manager* manages all *workers*, each of which use the Tor control protocol [193] to command and control the associated Tor client instance and its circuits. The workers run a modified Tor client instance, based on stable release `0.2.3.25`, that adds: a `STOPREADING` controller command which instructs Tor to stop reading from the onion routing connection to the target; `SENDSTREAMSENDME` and `SENDCIRCUITSENDME` commands which instructs Tor to send a stream-level and circuit-level `SENDME`s on the specified streams and circuits; and an `IGNOREPACKAGEWINDOW` command that instructs the client to ignore package windows when sending data upstream.

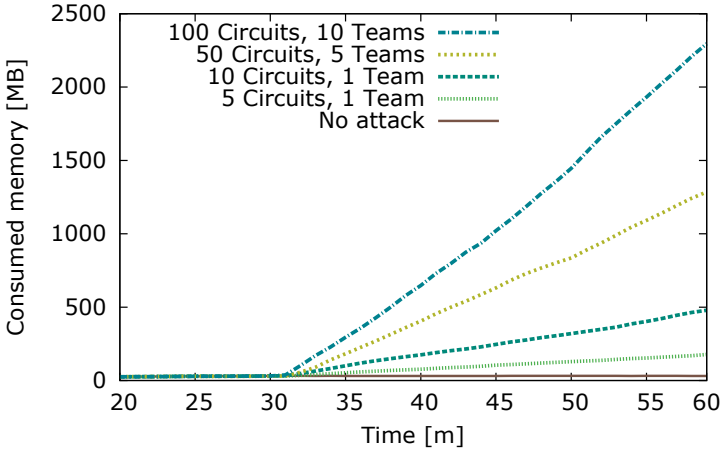
We implemented both direct and anonymous Sniper Attack modes. In direct mode, each worker connects to the Tor client over the controller port, waits for it to become fully bootstrapped into the Tor network, and builds its custom Tor circuits using the same path as the other workers on its team. Once the attack circuits are ready, the probing workers begin circuit measurement probes by downloading files through their attack circuit; the remaining workers request an extremely large file through the attack circuit, command Tor to stop reading, and send two stream `SENDME`s and one circuit `SENDME` for every completed probe download. In anonymous mode (see Section 5.2.3), each worker runs two Tor client instances instead of one: the first is used to create an anonymous tunnel through Tor; the second is used as in direct mode, except that all communication with relays is done over the anonymous tunnel using the provided SOCKS proxy interface. Note that the client instances that create the anonymous tunnels ignore their package windows using the `IGNOREPACKAGEWINDOW` command, because otherwise the `SENDME`s that are being forwarded from the attack circuits upstream through the tunnel will eventually drain the windows and stall the attack (the tunnel's normal downstream `SENDME`s which increment the package window will not be received because of the stop reading attack behavior). The sniper manager and worker logic was packaged as a Shadow plug-in consisting of 1416 lines of code, while our Tor modifications included 253 lines of code.

5.3.3 *Experiments and Results*

We experiment with our prototype implementation of the Sniper Attack to explore the target memory consumption and sniper resource tradeoffs when conducting the attack against target relays of various capacities. Our Tor network model is configured as described



(a) Target memory over time (direct attack mode).



(b) Target memory over time (anonymous attack mode).

Figure 24: The target relay's memory usage over time in direct and anonymous attack modes.

above, with the addition of an adversarial sniper node that runs the Tor clients and the sniper manager that controls the attack. Unless otherwise specified, our experiments use 100 circuits configured as 10 teams of 10 circuits each, while each probing circuit downloads $\sigma = 50\,\text{KiB}$ files, pausing for 60 seconds between each probe. Every team uses a unique Tor path for their circuits chosen by Tor's weighted path selection algorithm. Our sniper is configured with a 100 MiB/s symmetric bandwidth access link so as not to result in a bandwidth bottleneck during the experiment for measurement purposes. Each experiment runs for 60 minutes (simulation time), during the first 30 of which we allow the network to bootstrap and during the last 30 of which the attack is executed. We run one attack against a single target relay in each experiment, and measure the memory used over time by the target and the sniper as well as the bandwidth consumed by the sniper.

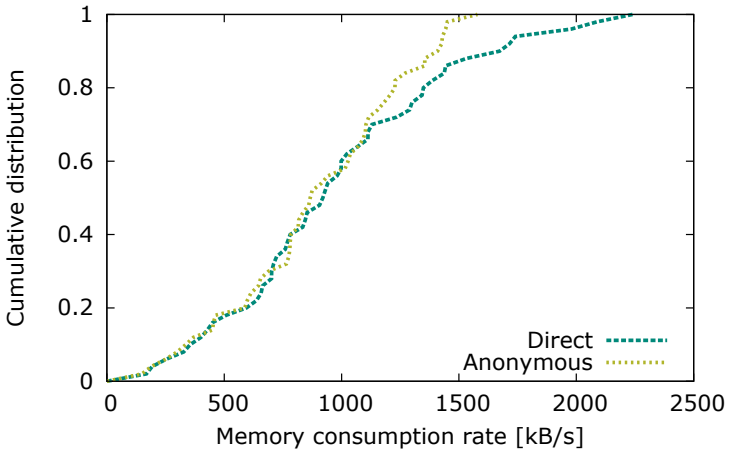
Setting		Direct			Anonymous		
		Mem	Tx	Rx	Mem	Tx	Rx
	100 Circs, 10 Teams	297	57	21	591	58	17
	50 Circs, 5 Teams	148	31	10	297	28	9
	10 Circs, 1 Teams	29	6	3	60	10	2
	5 Circs, 1 Team	29	4	2	59	4	2

Table 3: Sniper resource usage (memory in MB, mean tx/rx rates in kB/s).

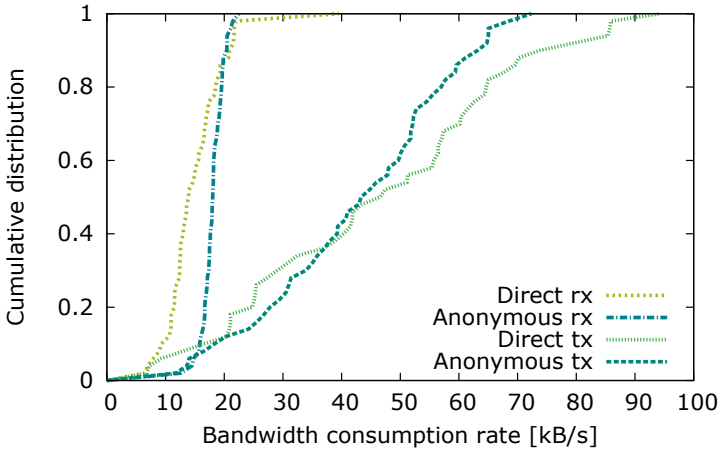
We tested the feasibility of the Sniper Attack, arbitrarily choosing the highest weighted non-authoritative, non-exit entry guard in our network as the target. This node had a 9 MiB/s symmetric bandwidth access link and otherwise served as an ordinary relay in our experiment. We tested the Sniper Attack with each of 100, 50, 10, and 5 attack circuits. As can be seen in Figure 24, the number of circuits directly affects the rate at which the sniper is able to consume the target's memory. While the target's memory consumed in each scenario increases approximately linearly, there is a dramatic difference between 10 and 50 circuits: the 10 circuit experiment was configured with 1 team, meaning that all 10 circuits are configured with the same path through Tor; the 50 circuit experiment was configured with 5 teams, meaning that there are 5 paths chosen through Tor. Choosing multiple paths in this way more effectively utilizes Tor's capacity and prevents the attack from stalling due to a poorly chosen circuit that may contain a tight bottleneck.

The memory and bandwidth requirements for the sniper in our feasibility experiments can be seen in Table 3. Shown are the maximum total memory (Mem in MB) used by the sniper at any point during the attacks and the mean total bandwidth consumption (tx/rx in kB/s), for both the direct and anonymous experiments. The memory used by the sniper depends almost entirely on the number of Tor instances being used: in all cases, the mean memory used per Tor client instance was approximately 15 MB. As expected, the anonymous attack consumes roughly twice as much memory as the direct attack since it is using twice as many Tor client instances. The resource requirements for our prototype are quite reasonable: the maximum memory required in any of our experiments was less than 630 MB and the maximum upstream and downstream bandwidth required was 58 and 21 kB/s, respectively. Further, the sniper's 60 second bandwidth burst remained below 500 kB/s throughout the experiment. We expect an adversary willing to launch this type of attack can easily satisfy these requirements. Note that probing less often may further reduce the bandwidth costs.

Our feasibility experiments tested the Sniper Attack against an arbitrarily chosen relay. We expanded this evaluation to determine how



(a) Target memory consumption



(b) Sniper bandwidth consumption

Figure 25: Sniper Attack resource consumption.

the Sniper Attack performs against a variety of relays with unique congestion, load, and bandwidth capacities. To do this, we chose a set of 50 relays from our private network, again using Tor's weighted path selection algorithm. Using the default settings outlined above, we ran our prototype Sniper Attack against each relay twice: once in direct mode and once in anonymous mode. We measured the memory consumed by the target and the bandwidth consumed by the sniper during each experiment.

We computed the mean target memory consumption rate and mean sniper bandwidth consumption rate achieved during each experiment (recall that each experiment targets a different relay). Figures 25a and 25b show the cumulative distribution of these rates for each mode over the 50 experiments; each experiment produces one data point in each of the two figures. As shown in Figure 25a, the median computed mean target memory consumption rate was 925 kB/s in the direct attack and 870 kB/s in the anonymous attack. Further,

the direct mode of the attack was only slightly more effective in our experiments: in the maximum the sniper was able to consume the target's memory at 2239 kB/s, roughly 1.4 times as fast as the maximum of 1579 kB/s in anonymous mode. Although this difference is only seen in the tail of the CDF, the reason is likely due to the additional length of the attack circuit path in anonymous mode (the cells must traverse 6 Tor relays in this case), which may lead to less accurate probing and extra latency when sending the `SENDME` cells through the anonymous Tor tunnel to the the opposite edge of the attack circuit. Further, the longer path increases the chance that a bottleneck exists on the path which may cause some of the attack circuits to fail. Figure 25b shows that the bandwidth requirements in both modes are similar: the mean upstream bandwidth measured was 47 and 44 kB/s in the median for the direct and anonymous attacks, while the mean downstream bandwidth was respectively 14 and 18 kB/s in the median. Our experiments show that the Sniper Attack enables the adversary to relatively easily trade its bandwidth resources for a victim relay's memory resources.

5.3.4 *Analysis*

We now analyze the practical effect the Sniper Attack has on the operational Tor network by considering how realistic adversaries might choose to disable relays. The adversary may prioritize as targets the relays with low memory but high consensus weights: this will have the largest impact on users since Tor's load balancing algorithm is tuned so that the probability that a client chooses a relay is in proportion to the bandwidth capacity that relay contributes to the network. However, since relay memory resources are not public, we consider an adversary that chooses relays based on the consensus weight alone and explore the time to disable them according to various potential memory configurations. Because of the load balancing algorithm and the fact that currently the relays with the top 100 weights constitute approximately 40 percent of the selection probability, the adversary may have significant impact on the network by disabling a relatively small group of relays.

We utilize the results from our 100 experiments discussed above to estimate memory consumption rates that an adversary may achieve on live Tor network relays. To do this, we compute the correlation between the observed mean memory consumption rate of each target relay in our experiments and that relay's consensus weight using a linear regression. This results in parameters that we use to estimate the memory consumption rate of any relay for which we have a consensus weight. Negative rate estimates were replaced with the minimum observed rate. We then use these rates to compute the time to disable various groups of relays: we consider the top, median, and bottom

			Direct		Anonymous	
		Sel %	1 GiB	8 GiB	1 GiB	8 GiB
Relay Groups	Top FAST Guard	1.7	0:01	0:18	0:02	0:14
	Median FAST Guard	0.025	0:23	3:07	0:23	3:07
	Bottom FAST Guard	1.9e-4	1:45	14:03	1:45	13:58
	Top FAST Exit	3.2	0:01	0:08	0:01	0:12
	Median FAST Exit	0.01	1:45	14:03	1:22	10:53
	Bottom FAST Exit	6e-5	1:45	14:03	1:48	14:20
	Top 5 Guards	6.5	0:08	1:03	0:12	1:37
	Top 20 Guards	19	0:45	5:58	1:07	8:56
	Top 5 Exits	13	0:05	0:37	0:07	0:57
	Top 20 Exits	35	0:29	3:50	0:44	5:52
	All Dir Auths	N/A	17:34	140:32	17:44	141:49

Table 4: Combined path selection probability of and expected time to disable selected groups of relays.

guard and exit relay by the probability of selection by clients out of those with the FAST flag, as relays without the FAST flag are only selected if no FAST relays are available. We also consider the top 5 and 20 of both guards and exits as those relays will be selected most often by clients and represent the most attractive targets for the adversary. We consider the 10 directory authorities as the final group, as the network will not function over time without the authoritative documents they collectively produce and distribute.

Shown in Table 4 is the total selection probability for each relay group, and the estimated total length of time to disable all relays in the group when the Sniper Attack is synchronously launched on a single relay at a time. We consider memory consumption rates for both direct and anonymous attacks, and consider the length of time to disable relays with 1 and 8 GiB of memory as examples of relay memory capacities. Note that these results scale linearly to other memory sizes. Also note that although the linear regression did not result in a strong correlation (direct: r^2=0.164, anonymous: r^2=0.237), we believe it provides a reasonable prediction of memory consumption for analysis purposes as we expect the actual time to disable the groups of relays given in Table 4 to fall somewhere the times given in the 1 GiB and 8 GiB columns.

Our analysis shows that the fastest guard and fastest exit with 1 GiB of memory can be disabled in just one minute when using the direct attack, thereby disabling an expected 1.7 and 3.5 percent of paths in the Tor network, respectively. When allotting 8 GiB of memory for these relays, they can be disabled in under 20 minutes in both attack modes. Perhaps more strikingly, the entire group of the fastest 20 ex-

its can be disabled in just 29 minutes if each relay has only 1 GiB of memory, and in just under 4 hours if each relay has 8 GiB of memory. The anonymous attack takes slightly longer in both cases. This would be extremely disruptive to the Tor network, causing roughly 35 percent of all paths to fail and increasing load and congestion on the remaining relays. Similarly, the group of the fastest 20 guards can be disabled in just 45 minutes if allotting 1 GiB of memory for each relay, and just under 6 hours if allotting 8 GiB of memory for each (again, the anonymous attack takes slightly longer). This would cause 19 percent of Tor paths to fail. Finally, the attack takes significantly longer on the group of directory authorities, since their lower bandwidth weights result in lower memory consumption rates than the fastest relay groups. Note that relays will likely be rebooted by their operators some time after going down, however, all circuits they were carrying will be lost and the attack could be relaunched against a relay as soon it is available. This may effectively cause a relay to be marked as unstable and not chosen by clients for their circuits.

5.4 DEANONYMIZATION IN TOR

The Sniper Attack is more than just a threat to Tor's availability: it can also be used to attack *anonymity*. Because Tor accepts any willing relay into the network, an adversary that runs relays can deanonymize a victim by controlling the entry and exit relays and correlating the observed timing and volume of a user's traffic entering the network with that leaving the network shortly afterwards [34, 188].

To prevent an adversary running relays from eventually being chosen for these positions, a user chooses a small set of entry guards, and begins all circuits at one of these guards. This protects the user from being directly observed as long as adversarial relays are not chosen as guards. Thus a user's guard set is an attractive target for the Sniper Attack. If few enough of a user's guards are responsive (at most 1 in Tor), the user will select new guards as replacements. By disabling the user's guards, the adversary can cause the user to choose new guards and hope that an adversarial relay is among them. This process can be repeated until the adversary succeeds.

This attack requires the adversary to identify the target's guards and to force her to choose new ones as soon as the old ones are disabled. Doing so is particularly easy with hidden services [156] because they create circuits on demand. Therefore, we will describe and analyze the attack applied to hidden services.

Deanonymizing Tor clients using the Sniper Attack is less straightforward because they generally do not respond on demand. However, in some significant cases guards could be identified and guard reselection initiated. For example, a user downloading a large file could give the adversary enough time to discover the guard using a con-

gestion side channel [78, 89, 146]. Furthermore, download managers and BitTorrent clients generally automatically restart an interrupted download, which would prompt guard reselection by Tor.

Finally, we note that in addition to deanonymization, the adversary could use the Sniper Attack to attack Tor privacy in other ways. For example, he could attack the exits of long-lived circuits, such as IRC connections, in order to be chosen as the replacement exit and discover the destination. He could also attack exit relays that allow connections to certain ports in order for adversarial relays to observe a larger fraction of exit traffic to such ports.

5.5 DEFENSES AGAINST SNIPER ATTACKS

The Sniper Attack exploits fundamental problems with Tor's design. In Chapter 7 we will present our tailored transport protocol for Tor, which, besides improving the performance, also protects from the Sniper Attack's attack vector by allowing Tor to drop cells. While a redesign is due to many reasons the preferable approach, it is still necessary to develop a fix against the Sniper Attack for the operational Tor network. In this section, we therefore explore defense strategies (summarized in Table 5) and their costs, limitations, and practical operational deployment issues.

5.5.1 *Authenticated* `SENDME`*s*

One problem exploited by the Sniper Attack is that the packaging edges are unable to verify that the delivery edges actually received any cells. One solution to this problem is adding a challenge-response puzzle to every 100th cell. Each packaged cell currently includes a hash digest of its contents so that bit errors may be detected by the client. A package edge can require that the digest of each packaged cell be included in the corresponding `SENDME` feedback signal cell. To prevent the delivery edge from pre-computing this digest when downloading a known file, the package edge could include a 1 byte nonce in every 100th cell. This nonce will randomize the digest that must be returned in the `SENDME`, and can only be guessed with probability 1/256. If the response digest doesn't match the challenge, the exit can drop the circuit. Authenticated `SENDME`s prevent clients from subverting the 1000 cell in-flight limit, including those who attempt to "cheat" by preemptively sending `SENDME`s to the exit in order to download data faster.

This defense provides an elegant solution to detecting protocol violations. It defends against a single client using the efficient version of the Sniper Attack. However, using this approach alone has some limitations. First, it does not completely stop the attack: each circuit will still be able to cause the target to queue 1000 cells (512 kB), and so the target can still be taken down using the parallel attack. Second, relays

		Defeats the Sniper Attack		
		Basic	Efficient	Parallel
Defense	Authentication	○	●	○
	Length limit	●	●	○
	Circuit killer	●	●	●

Table 5: Defense capabilities.

are relying on honest circuit members to perform the authentication protocol correctly, and therefore this defense does not protect against the basic version of the Sniper Attack where the packaging edge is malicious. We could improve the situation by allowing intermediate relays to read and detect unexpected `SENDME` cells and destroy the circuit, but we note that a self-defense strategy is preferred to one that relies on other circuit members. Finally, this approach has a longer transition phase, since all clients and at least all exit relays need to be aware of the authentication protocol.

5.5.2 *Queue Length Limit*

Another problem exploited by the Sniper Attack is that Tor's application queues may grow without interference by the relay. Therefore, a simple defense is for each relay to enforce a maximum queue size to limit the amount of memory each circuit may consume. If the queue length becomes greater than the allowed size, then the relay may assume a protocol violation and destroy the circuit to inhibit malicious activities.

To find a good candidate queue size, we consider that Tor's flow control algorithm already enforces a limit on the number of cells that may be in transit (1000, plus some tolerance for control messages). One approach would be to use a similar limit as a queue length limit, which provides a self-defense mechanism while also protecting against adversaries who control multiple nodes in a circuit. However, as with the authenticated `SENDME`s defense, a queue length limit does not prevent an adversary that uses the parallel Sniper Attack from circumventing the memory limitations, since memory consumption from its multiple circuits in aggregate can still crash the relay with relatively low overhead. Further, a maximum queue length would obstruct future development. Considering that we work towards a tailored transport protocol with a dynamic feedback mechanisms, a hard threshold on the queue length may complicate migrations. Finally, we note that the queue length limit defense enables a new attack in which web servers could inject page objects that require new streams and cause benign circuit queues to grow beyond the limit and therefore be destroyed [67].

5.5.3 *Adaptive Circuit Killing*

To overcome the limitations of the previous defenses and protect against the parallel Sniper Attack, we now develop a more sophisticated, adaptive mechanism which is incrementally deployable and has strong security properties. A clever attacker against both of the previous defenses can use a sufficiently high number of parallel circuits, each with a short queue, to exhaust a relay's memory. To prevent memory exhaustion, a relay can begin and continue to kill circuits while the *total* memory consumption remains above a critical memory threshold. This technique will guarantee that a relay process will not terminate due to an out-of-memory condition.

SELECTING CIRCUITS The central question to be solved is to decide which circuit should be killed if memory becomes scarce. This question is not as simple to answer as it might seem at a first glance. For instance, the most straightforward approach would be to kill the circuit with the longest queue. This, however, can be leveraged for a new attack: an adversary could set up a large number of circuits with relatively short queues on a given relay, so that this relay's memory consumption is very close to critical. Whenever a benign circuit temporarily builds up a long queue, the threshold will be exceeded and a *benign* circuit will be killed, while the adversary's (shorter) circuits will remain in place. The relay is therefore manipulated in such a way that it will regularly kill benign circuits—without any need for the attacker to spend resources beyond initially setting up the circuits. While the relay will not crash due to running out of memory, this is still highly undesirable.

We must therefore aim for a decision criterion which cannot be abused by an attacker to make a relay kill benign circuits. Here, we propose to use the time of arrival of the frontmost cell in the queue as the basis for our decision: if memory becomes scarce, the circuit killing mechanism will kill the circuit with the currently "oldest" cell at the front of its queue. We require that each incoming cell be tagged with a timestamp upon arrival at a relay, but note that this already happens in the current versions of Tor in order to compute cell delay statistics. Therefore, this mechanism is almost trivial to implement. In the remainder of this section, we will argue why it is also effective.

To gain an intuitive understanding, observe that an attacker—in order to avoid that his circuit is killed when memory becomes scarce—will have to keep the frontmost cell in the circuit's queue "fresh". Since Tor circuit queues are strict FIFO queues, the frontmost cell in any given circuit queue will have spent more time in this queue than any other cell. The attacker is therefore forced to continuously read from all his circuits; otherwise, the cell at the attack circuit's head will soon be older than the frontmost cells in the queues of benign circuits.

Thus, by deriving bounds on the share of the relay's available bandwidth that is required in order to make a relay kill a benign circuit, we will be able to prove the effectiveness of the defense strategy.

PROOF SKETCH Consider a specific relay which offers a total bandwidth B for relaying Tor circuits. We assume that B is available both in incoming and in outgoing direction (substantially imbalanced incoming and outgoing bandwidths do not make sense for a relay which essentially forwards all incoming data). Furthermore, assume that this relay is currently used by a total of n active circuits. We define an active circuit as a circuit which currently has at least one cell in its queue.

If the outgoing bandwidth of the relay were assigned to the active circuits in a perfectly fair manner, then each circuit would experience an outgoing data rate of

$$r_{fair} = \frac{B}{n}.$$

Of course, in practice, the distribution will not be perfectly fair. In fact, there are certain issues with respect to inter-circuit fairness, which we will address in the following chapter. While these issues lead to gross unfairness between circuits, Tor relays include mechanisms which will still yield circuit bandwidth allocations that are not arbitrarily unfair: there is a round-robin scheduler which picks cells from circuit queues for transmission. Moreover, as we know from Chapter 3, circuits are carried over TCP connections, and TCP, too, strives for a fair distribution of available bandwidth to multiple connections. Both of these mechanisms are controlled by the relay and are thus outside the sphere of influence of an attacker. We will discuss the case of an attacker who is able to claim a huge fraction of the relay bandwidth for himself later. For now, we may reasonably assume that there is a fairness factor $0 < \alpha \leqslant 1$ such that each active circuit receives a bandwidth share of at least

$$r \geqslant \alpha \frac{B}{n}. \tag{9}$$

As we will see, the exact value of α is not critical for our scheme, as long as an active circuit's bandwidth share does not become arbitrarily small for a longer period of time.

Now observe that benign circuits will typically have queues which are bounded above by a relatively small size Q. Q is, as discussed before, in the order of 1000 cells in the current Tor protocol. Even if possible future protocol versions do not enforce a hard upper limit, observe that high values of Q imply long queues in the relays and thus poor circuit performance. In practice, any reasonable future protocol design will therefore also result in reasonable queue lengths.

Note that while we assume that such an upper bound Q exists in our analysis, its value need not be known and is not used to decide which circuit to kill. The exact value is thus much less critical than in the previously discussed queue length defense.

Based on these assumptions we make a central observation for our argument: if a benign circuit's queue length does not exceed Q and its mean rate is at least r, then the maximum time for which a cell can remain queued is bounded above by

$$d_{max} = \frac{Q}{r} = \frac{Qn}{\alpha B}.$$

Therefore, if t_{now} is the current point in time, the cells at the heads of all benign circuits' queues will have a timestamp later than $t_{now} - d_{max}$.

Note that an attacker using a single circuit will thus have to make sure that the cell at the front of the queue does not become older than d_{max}, i. e., the cell must have arrived at a point in time later than $t_{now} - d_{max}$. Only then can the attacker hope that a benign circuit will be killed instead of the attacker's circuit. If the attacker uses multiple circuits in parallel, the same criterion must hold for all these circuits. Consequently, all the cells in the attacker's circuits must have arrived within a time interval of length d_{max}.

Let the amount of free memory at the relay be denoted by M. The attacker must (roughly) build up queues with a total size of M bytes in order to make the relay kill circuits. Since, as seen before, the attacker must inject all these cells within a time span of length d_{max}, the attacker needs to send cells into the relay at a mean rate of at least

$$r_a = \frac{M}{d_{max}} = \frac{M}{Q} \cdot \alpha \frac{B}{n} = \frac{M}{Q} \cdot r.$$

This is a factor of M/Q higher than the minimum outgoing rate r which we assumed for benign circuits above in (9). Observe that M/Q can easily be made a very large number if sufficient memory is provided. We recommend an order of magnitude of a few hundred megabytes, which is not a problem on today's relays (also on machines with a 32 bit address space) and results in a factor M/Q in the order of 1000.

The attacker would therefore have to claim the incoming relay bandwidth virtually entirely for himself in order to mount a successful attack that results in a benign circuit being killed. Although such an attack is possible if an adversary has enough bandwidth, we consider it practically unrealistic for two key reasons: first, fairness mechanisms are in place also on the incoming side of a relay, making it very hard to achieve this in the first place; and second (and much more important), observe that consuming almost all of a relay's bandwidth constitutes by itself a far more devastating attack on the relay.

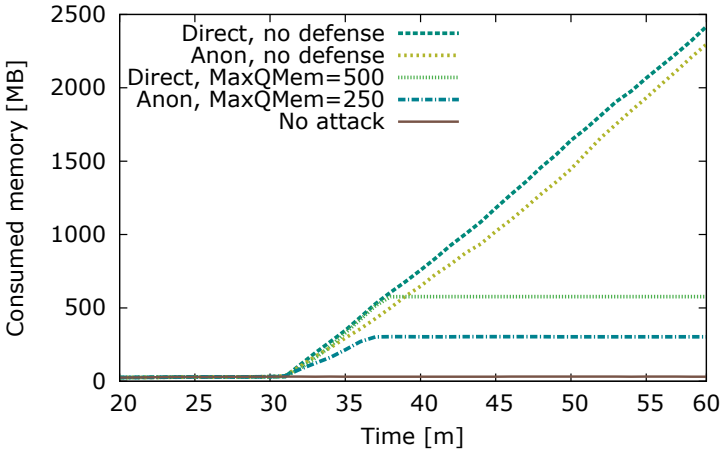


Figure 26: The circuit killer renders the Sniper Attack ineffective.

An adversary with enough bandwidth to succeed in this attack and cause a relay to drop a few benign circuits would do more damage using its bandwidth in a classic DoS attack, or in a selective DoS attack [43] launched while running malicious relays. (A bandwidth attack on a relay may in fact kill benign circuits anyway, e. g., due to TCP connections timing out.)

EVALUATION We implemented the described out-of-memory (oom) circuit killing as a Tor software patch. It introduces a new configuration parameter `MaxQMem`, which specifies the maximum amount of memory usable by Tor's circuit queues. Every second the algorithm checks for violations to this threshold and kills a circuit if necessary. We re-run the experiments from Section 5.3 with the oom circuit killer deployed on all relays, using a `MaxQMem` of 500 MiB for the direct and 250 MiB for the anonymous Sniper Attack (we chose different values solely for a clearer presentation). The results in Figure 26 contrast the memory consumption with and without our defense. With our defense in place, it depicts a horizontal line around the configured `MaxQMem` during the attack, showing that the consumed memory is bounded by our new parameter. Closer examination shows a microscopic oscillation around the threshold, i.e. first surpassing it, then freeing memory, and then rising again due to the other sniper circuits. It successfully protects the process from arbitrarily increasing the memory consumption and thus from being killed. During the experiments of the direct and the anonymous attack the circuit killer intervened 43 and 32 times respectively, and in all cases only attacking circuits were killed. Thus this defense resulted in a 100% identification rate with no false positives.

The above results reveal insights into the interplay between fairness and the robustness against the Sniper Attack when such a mechanism is in place. An attacker needs a lower rate and thus fewer resources

either if the queues of benign circuits become longer (higher value of Q) or if the distribution of relay bandwidth to the circuits becomes less fair (smaller value of α). Approaches that improve the fairness of transmission scheduling in Tor (as we present in Chapter 6), or bound the queue lengths based on hop-by-hop feedback (as we present in Chapter 7) would therefore complement this defense strategy.

In summary, we believe that adaptive circuit killing based on the queuing duration of the cells at the heads of the queues constitutes a strong defense. Not only does it prevent a relay from crashing due to insufficient memory, it is also very resilient against being abused to make relays kill benign circuits. It is simple to implement and easily deployable: the mechanism need only be implemented on the relays, and it is immediately effective on all relays where it is deployed.

Our proposed defenses against the Sniper Attack protect against memory exhaustion but do not protect against brute force network or CPU overloading. In addition, other DoS attacks on Tor continue to be discovered [139, 159], and a Tor relay is vulnerable to all DoS attacks on the host platform. The Deanonymization DoS Attack can be performed using *any* DoS attack on a Tor relay and thus is still a serious problem.

As a defense against it, we suggest that the Tor client limit the number of relays that it chooses for the most sensitive positions in its circuits. In the following we describe this proposal in detail, and we evaluate its security and its cost in terms of network performance.

5.6 RELATED WORK

Internet DoS attacks, those that make an Internet service unavailable for longer than the intended waiting time [206], have been extensively studied in the literature. Although unique in this space, the Sniper Attack is most closely related to *low rate* and *slow read* DoS attacks, which are variants of the well-known SYN flood attack [75, 181]. The goal of these attacks is to exhaust resources in order to prevent the victim from processing new incoming connection requests. Transport layer low rate attacks [121] exploit TCP's retransmission timeout (RTO) dynamics. An attacker repeatedly sends short high-rate packet bursts, which produce packet losses (i. e., timeouts) and thus make the victim double the RTO of other TCP connections [161]. Transport layer slow read attacks [105] send legitimate data requests, advertise a small TCP receive window, and then slowly empty the receive buffer. As a result, the victim's send buffer remains full over a long time span, thus blocking resources. Similar low rate and slow read techniques have been described to exploit web server weaknesses on the application layer [45, 173, 174]: sending partial HTTP requests or slowly reading responses to requests will prolong HTTP sessions and

extends the time in which the availability of the web server's connection pool is reduced.

Although the Sniper Attack shares the general goal of preventing new incoming connections with low rate and slow read attacks, it is achieved as a byproduct of the more direct goal of exhausting system memory resources. In particular, we consume memory from the application layer using valid overlay network protocol messages without reading from the victim. Therefore, our attack may be characterized as a *no read* attack. Another important distinction is that, unlike the attacks described above, our attack does not require several simultaneous connections to the target and continued effort in order to maintain the effect of the attack. Finally, our attack destroys existing established connections in addition to preventing new ones.

The Sniper Attack may also be categorized as a *permanent* DoS attack, as it exploits application layer overlay network protocol semantics to consume system memory and crash the process. It is distinguished from similar attacks, such as the Ping of Death [114], in that it utilizes valid messages to exploit the protocol design. Fixing it is therefore not simply a matter of correcting a broken protocol implementation.

Our attack is also similar to those that rely on misbehaving receivers and optimistic `ACK`s to bypass flow control mechanisms [14, 176, 183]. In particular, the opt-`ACK` attack [183] is similarly challenged to adjust a feedback signal rate in such a way that it still appears legitimate to the communication partner. Our attack differs in that we target application layer protocols of overlay networks in order to exhaust the available memory, rather than targeting network layer protocols for the purposes of consuming the available bandwidth. As such, the Sniper Attack is a *no read memory exhaustion* attack.

DoS attacks against the Tor overlay network have been studied before, building upon a fundamental observation first made by Syverson et al. [188]: if the first and the last relay along a Tor path are compromised, an adversary can link the source and destination by correlating traffic patterns. Øverlier and Syverson first demonstrated how an adversary could lie about the available bandwidth of compromised relays in order to inflate the probability of being selected for a hidden service circuit [156], and Bauer et al. extended the attack to increase the probability of end-to-end compromise of general purpose circuits [34]. Borisov et al. [43] describe a selective DoS attack on Tor where malicious relays terminate circuits of which they are a part but do not control both ends. This forces clients to re-build circuits and similarly increases the probability of end-to-end compromise by the adversary. Danner et al. show how selective DoS attacks can be provably detected by exhaustively probing potential paths [62], while Das and Borisov reduce the cost of detection using probabilistic inference [63].

Resource consumption attacks that may also be used to increase an adversary's probability of end-to-end circuit compromise include the Packet Spinning attack [159] and the CellFlood attack [139]. In the Packet Spinning attack, the adversary crafts special packets that cause them to continuously "spin" through circular circuits composed of the target relays. In the CellFlood attack, the adversary uses special handshake packets to efficiently build a large number of circuits through the target relays. Both of these attacks effectively make relays appear busy by forcing them to spend resources doing unnecessary work. Honest clients' circuits through these relays will then be more likely to time out, causing them to choose new circuits containing malicious relays with higher probability. The Sniper Attack also causes relays to perform unnecessary work, but focuses on consuming memory resource rather than bandwidth or computational resources.

5.7 CHAPTER SUMMARY

In this chapter we presented a novel and destructive DoS attack against Tor that may be used to disable arbitrary Tor relays by exploiting the protocol's reliable end-to-end data transport. We outlined several ways to carry out the Sniper Attack and assessed its resource and time profiles in large scale simulations. We performed an in-depth security analysis and developed a defense that renders the attack ineffective by identifying and killing malicious circuits in out-of-memory situations.

Although the Sniper Attack is tuned for Tor, our mechanisms may generalize to systems that do hop-by-hop reliability and end-to-end flow control. This key insight contributes to the design specification of our custom transport protocol. In particular, it confirms our initial assertion that instead of dedicated end-to-end flow control a backpressure-based approach is the favorable choice. We also note, that the Tor application should be in control of reliability, i. e., they should be able to drop cells. Again, this confirms our design decision to implement TCP-semantics on the application layer and to tunnel our protocol over UDP.

Finally, although our defenses prevent memory exhaustion in the Tor network, they do not stop the Sniper Attack from consuming a large amount of Tor's bandwidth capacity at low cost.

6 DYNAMIC RESOURCE ALLOCATION

6.1 OVERVIEW

This chapter is based on previous work by the author [10, 11].

Mechanisms for handling congestion and fairness in anonymity networks, where user privacy is of greatest significance, are not yet well understood. Thus, current designs leave a lot to be desired: gross unfairness and largely suboptimal performance can be observed. As virtually all the previous chapters indicated, though, fairness is, both from the security and performance perspective, a main component to build a resilient network.

In this chapter, we focus on fairness aspects between circuits. We first show that interactions of multiple scheduling mechanisms in the current Tor design cause heavily unfair resource allocations to users. To this end, we first analyze the current scheduling mechanisms in Tor theoretically and argue why they lead to unfairness even in very simple settings. To complement our arguments with a second perspective, we also demonstrate the problems based on packet-level network simulations.

Subsequently, we develop a fairness model based on *max-min fairness* that takes the specifics of anonymity networks into account. This leads us to a re-design of Tor's scheduling. Our scheduling approach overcomes the unfairness problems which are exhibited by today's Tor implementation. It achieves global max-min fairness and thus a fair resource allocation despite selfish end-users.

Yet, the results make Tor's fundamental performance problem imminent: due to the fixed-size end-to-end sliding window mechanism excessive queues build up in front of bottlenecks. Our new scheduler design shifts the queues from the transport layer to the application layer, which is necessary to take full control of the packet scheduling. Thus, the approach is a key contribution (which we will use in the following chapter) to tackle the root cause for the prevalent performance issues.

6.2 FAIRNESS IN AN ANONYMITY OVERLAY

In Tor, relay operators donate resources, in particular bandwidth, to the anonymity overlay. Each operator of a relay can configure a bandwidth limit. The relay is not allowed to use more bandwidth than this limit. Therefore, the operator can control how much resources she wants to contribute, and can ensure not to exceed limits that exist, e. g., due to provider contracts. Typically, the bandwidth limit config-

ured in Tor is lower than the bandwidth that is physically available to the relay. In fact, this is an important assumption implicitly made in the design of Tor: the bottlenecks in today's Internet are usually not in the network backbone, but at the access links. If the configured limit is below the respective access link capacities, TCP congestion control and fairness will therefore not interfere with the bandwidth allocation determined by Tor. Relay operators shall therefore take care to configure a not-too-large bandwidth limit. Here, we are primarily interested in the scheduling mechanisms within Tor and thus make the same assumption.

But how should the resources of relays be allocated to the individual connections and circuits? At a first glance, this sounds as if standard solutions from the area of congestion control and fairness are readily applicable. However, there are at least two aspects that make the situation considered here subtly, yet significantly different. First, introducing additional signaling in Tor is to be avoided. Low signaling overhead is generally a desirable trait, in order to keep the control overhead as low as possible. However, in an anonymity system like Tor exchanging as little explicit control information as possible is particularly crucial also for another reason: control data is always at the risk of leaking information that can be used to break the anonymity. We therefore aim at a scheduling algorithm that works without additional control messages, based on local operations only.

Second, it must be taken into account that we are dealing with an overlay network. In a physical network like the Internet, each outgoing link of a router is a separate, independent "channel" to one neighboring node. The capacities of these links can thus independently be used (or remain unused). As we already pointed out before, the links between relays in Tor, in contrast, are not independent: all outgoing connections will typically use the same Internet access link, and all of them must share the configured bandwidth limit. This total available bandwidth can thus be freely allocated to individual overlay links—which, on the one hand, provides additional degrees of freedom, but on the other hand also requires a very careful design with respect to fairness.

As we will see, the mentioned differences to standard settings in computer networking have a notable impact on the fairness model. In the following, we will therefore first look at the current scheduling and fairness mechanisms in Tor, and point out major deficiencies in the existing design. Subsequently, we introduce a suitable fairness model and discuss how it can be introduced into Tor.

6.3 FUNDAMENTAL FAIRNESS ISSUES IN TOR

We now turn towards the specific mechanisms in Tor that are currently used for scheduling the transfer of cells, and point out where the key deficiencies of the status quo are.

6.3.1 *Scheduling and Multiplexing*

Buffering and scheduling in Tor is organized in a nested, hierarchical structure. The most important distinction made in this context is between connections and circuits: connections are TCP/TLS links between neighboring relays, circuits are the logical end-to-end user connections through this overlay. A circuit thus traverses multiple overlay hops, and thus multiple connections.

The overall structure of the cell scheduling in a Tor relay is illustrated in Figure 27. Relays associate every TCP buffer with an internal connection buffer. Incoming data is fetched from the socket's incoming buffer and moved to the corresponding connection input buffer. In between these two buffers, a round robin and token bucket mechanism coordinates the fair distribution of capacity among connections. The configured rate and bandwidth determines how rapidly the token bucket is refilled and its maximum burst size. The round robin scheduler distributes the available tokens in the token bucket evenly among all incoming connections with pending data in their socket buffer. The scheduler is implemented implicitly, by scheduling read events via the used event library [128].

On the outgoing side, a mirrored construction exists and works identically: there is a second, independent token bucket which is refilled at the same rate and limits the outgoing traffic, and there is a second, again implicitly implemented round robin scheduling mechanism. It distributes the configured maximum bandwidth evenly across the outgoing connections.

Immediately after they have arrived in a connection input buffer, cells are rearranged into their respective circuit queues (in FIFO order), where they wait for further processing. This is also where the cryptographic operations for ensuring anonymity are performed. As previous work has shown, these operations are not a performance bottleneck [166]. They can therefore be safely neglected here.

Since each circuit travels through the overlay along a fixed path, the cells within one circuit queue are all forwarded to the same outgoing connection. This may be an overlay connection to another Tor node, or a connection from an exit router to an anonymously contacted host. As soon as the fill level of a connection output buffer falls be-

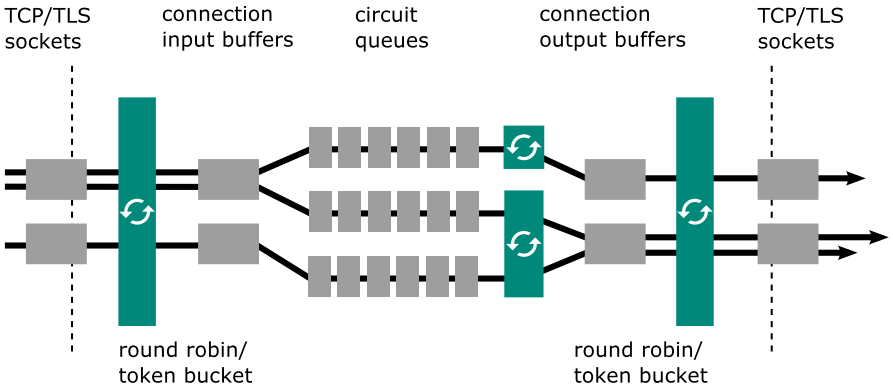


Figure 27: Scheduling and multiplexing in Tor.

low a certain low watermark, it fetches cells one-by-one from all the circuits traveling through it in a round robin manner. The scheduling algorithm assigns the same bandwidth to all active circuits traversing the connection.

6.3.2 *Token Bucket Issues*

The specific usage and implementation of the token bucket algorithm is one cause for very high (and unnecessary) queuing times in relays. We observe that token buckets in Tor are (surprisingly at a first glance) allowed to take on negative fill levels. This is justified by the TLS connections between relays where whole TLS records need to be processed. The token bucket on the incoming side often runs into non-negligible negative fill levels. As a consequence of this behavior, sometimes slightly more data is read than it would be admissible upon strict interpretation of the token bucket concept.

However, the token bucket for limiting the outgoing rate does not take on negative fill levels in the same way. Consequently, it regularly happens that somewhat more data are read on the incoming side than the outgoing token bucket allows to be written during the same cycle, even if their configured data rates are the same. The respective cells will thus not be allowed to leave the relay immediately. They will necessarily be queued for at least as long as it takes until the token bucket on the outgoing side is refilled again. One could say that the two buckets, on the incoming and outgoing side, work like a double door system and frequently lock cells for a full token bucket refill interval length (originally one second).

Additionally a coarse-grained refill interval of the token buckets has other detrimental effects. First, consider a relay with multiple TLS connections over which cells arrive. If there is high activity (i. e., many incoming cells in total), then the coarse refill interval will cause unfairness: it can happen that just because cells arrive at the "wrong" point in time, they must wait. This results in unnecessary, long queu-

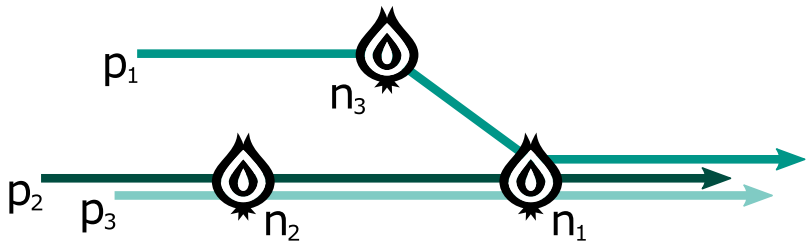


Figure 28: Simple unfairness scenario in Tor.

ing delays in the incoming socket buffers. These delays are in *addition* to the above discussed queuing delays in the circuit queues.

Second, the coarse-grained refill intervals also result in very bursty outgoing traffic pattern at relays. This is undesirable, since such a traffic pattern can interfere with TCP's control mechanisms and can be the source of suboptimal TCP performance on the TLS links between relays.

While the developed means in the remainder of this chapter overcome the "double door effect", we also proposed "stand-alone" remedies for the Tor software [4, 12] (of which one is already deployed). The proposals are very simple to implement, but nevertheless a significant reduction of cell queuing times can be expected [68]. Moreover, an incremental deployment is possible. As more relays upgrade, gradual performance improvements can be expected.

In the following, we will turn to more difficile fairness issues, which are by far not trivial to fix.

6.3.3 *Fairness Issues*

From the above analysis of the scheduling in Tor, it becomes clear that there are a total of three scheduling mechanisms, nested in two levels: on the connection level on both the incoming and the outgoing side of a relay, and on the circuit level within each outgoing connection. They are intended to apply fairness on the connection and circuit level, respectively. However, a closer look reveals that this does *not* result in a overall fair bandwidth allocation to circuits, not even within one single relay. In order to illustrate this, we first point out a very fundamental defect that can lead to gross unfair bandwidth allocations even in very simple settings, before we subsequently turn towards a more generic perspective.

This key defect results from the independence of the nested schedulers on the connection and circuit levels: each connection is assigned the same bandwidth b, and each connection's bandwidth is independently subdivided into equal per-circuit shares—regardless of the number of circuits traversing a particular connection. The rate implicitly allocated to a given circuit can therefore differ drastically, depending on the number of circuits per connection.

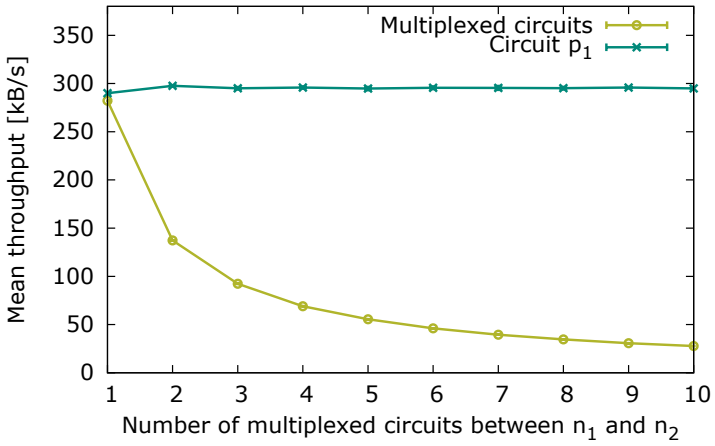


Figure 29: Unfairness in Tor.

A simple example, sketched in Figure 28, illustrates this unfairness problem. Assume that there are three circuits p_1, p_2, and p_3. All three pass through the same relay n_1. Further assume that n_1 is saturated, i. e., the circuits produce incoming data at a rate that exceeds the relay's configured maximum bandwidth, in the example 600 kB/s. As explained before, both incoming connections— one from n_2 and one from n_3—will get an equal (supposedly "fair") share of the bandwidth, in this case 300 kB/s. The circuits will share the respective connection capacity evenly. Therefore circuits p_2 and p_3, which share the same incoming connection, get 150 kB/s each, whereas p_3 gets 300 kB/s.

In order to be able to assess the behavior of Tor scheduling and of proposed modifications, we have implemented a simulation model of Tor nodes in the packet-level network simulator ns-3 [102]. We will later describe and discuss this simulation model of a Tor overlay in more detail. For now, we use it to underline that the expected unfairness indeed occurs if scheduling is performed as currently in Tor. To this end, we set up an overlay as in Figure 28, but we varied the number of multiplexed circuits on the connection between n_1 and n_2. Figure 29 shows the throughput obtained by the single circuit from n_3 and the mean throughput of the multiplexed circuits on the connection from n_2. The results in the figure are means over multiple simulation runs; throughout this chapter, error bars show 95 % confidence intervals.

If there is only one single circuit per connection, then only the connection-level scheduler has an impact. In that case, it may be expected that both circuits get an equal bandwidth share; the results show that this is indeed the case. For two or more multiplexed circuits the connection to p_1 gets a constant throughput of about half the available capacity, so that each multiplexed circuit receives a smaller

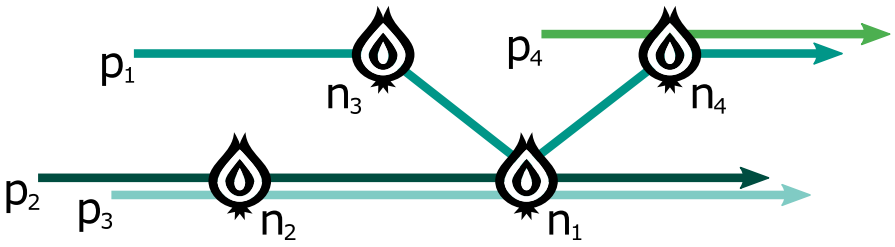


Figure 30: Max-min fairness example.

and smaller—and, in comparison to the single circuit p_1 increasingly unfair—fraction of the available capacity.

Similar problems exist analogously on the outgoing side, where cells from connections over which many circuits are multiplexed are written at the same rate as connections traversed by only one or very few circuits. These fairness issues are not as straightforward to address as it may seem at a first glance, since not all circuits are always active: the activity of a circuit depends on whether the corresponding user currently transmits data or not. It does therefore not suffice to just increase the bandwidth allocated to a connection according to the number of circuits currently established over it.

6.4 MAX-MIN OVERLAY SCHEDULING

The unfairness effects in the current Tor scheduling lead us to the central question in this chapter: how to design a scheduling mechanism which achieves fairness between circuits? A naive answer would be to uniformly allocate bandwidth to all currently active circuits. However, consider the example in Figure 30. As before, three circuits pass through a relay n_1. In n_4, one of the circuits "meets" another one. Let us assume that all relays have the same bandwidth limit (600 kB/s) and that there are no bottlenecks except the ones depicted here. If we allocated the same share of bandwidth (200 kB/s) to all circuits following an equal sharing policy, the result would indeed be fair—but it would also needlessly waste significant capacity in n_4, where 200 kB/s would remain unused.

A better solution would be to allocate this leftover portion of n_4's bandwidth to circuit p_4, so that this circuit is allowed to transport 400 kB/s in total. By doing so, we fully utilize the available resources without disadvantaging any other circuit. This approach follows the principles of *max-min fairness* [38]. With max-min fairness, flows with unsatisfied demands get an equal share of the available bandwidth, but never more than their demand. This is achieved by iteratively maximizing the allocated minimum while respecting the fairness criteria.

In the following, we will capture these abstract and paraphrased statements in a more general, formal, and rigorous way. In the con-

text of circuit switched networks, max-min fairness is a well-known allocation scheme. As prime example one can consider the available bit rate (ABR) service model in ATM networks [112, 148]. Tor, though, is a special case in several regards, and existing techniques and models are not readily applicable in the anonymity context, especially if excessive control and coordination overhead are to be avoided. Adaptations to both the overlay nature of Tor and to the specific requirements of an anonymity service are among the key contributions of this work. We will show that with a careful design of the scheduler, a surprisingly simple mechanism yields excellent results.

As a first step towards such a scheduler, we now review the concept of max-min fairness for standard wireline networks with dedicated links. We then discuss the necessary modifications for applying it in an overlay network, before we finally turn towards the question how to achieve max-min fairness between circuits in the Tor overlay.

6.4.1 *Max-min Fairness*

Consider a network, represented by a graph $G(N, L)$ where N is a set of nodes and L is a set of (physical, independent) network links between these nodes. Each individual link $l \in L$ has a limited bandwidth $C_l \geqslant 0$. P denotes the set of end-to-end connections through the network.

For the moment, assume that each connection is interested in maximizing its allocated rate, i. e., each connection has an arbitrarily large amount of data waiting to be transferred. Now let $r = (r_1, \ldots, r_P)$ be the vector of rates allocated to the individual connections in P. In the context of max-min fairness, such a *rate vector* is called *feasible* if all r_p are non-negative and for each link $l \in L$ it holds that

$$\sum_{i \in P_l} r_i \leqslant C_l,$$

where P_l denotes the subset of connections from P which traverse link l. A max-min fair rate vector is then defined as follows in [38]:

DEFINITION 1 A feasible rate vector r is called *max-min fair* iff

$$\forall p \in P \text{ and } \forall r' = (r'_1, \ldots, r'_P) \text{ feasible with } r_p \leqslant r'_p :$$
$$\exists p' \in P \text{ with } r_p \geqslant r_{p'} \text{ and } r_{p'} \geqslant r'_{p'}.$$

In essence, this definition says that the rate r_p allocated to a connection p cannot be increased without either violating the feasibility of r or reducing the rate of another connection p' which has a rate $r_{p'}$ that is already smaller than r_p. In terms of the whole vector r, we do not find another rate allocation r' that achieves higher rates without claiming resources of "smaller" connections. This matches our

intuitive solution from the example above. Thus, in a system which allocates rates in a max-min fair way, selfish users cannot exceed their fair share of resources, but at the same time no resources are wasted.

If a connection does not currently want to make full use of its bandwidth share, the leftover bandwidth may of course be allocated to other connections, again according to the above criteria, in a straightforward manner.

6.4.2 *Max-min Fairness in Overlay Networks*

In an overlay network like Tor, the above standard definition of max-min fairness is not applicable. We need to consider the special characteristics of such networks: Let $G(N, E)$ now be the (fully meshed) Tor overlay network. Thus, E is now a set of *overlay links*. In analogy to above, let P now denote the set of Tor circuits. Furthermore, let P_n denote the subset of circuits traversing relay $n \in N$.

Due to the fact that overlay nodes are end systems with only one network interface, all overlay links into or out of a given node n share the same physical link and, in sum, must respect n's configured bandwidth limit. We therefore do not have per-link bandwidth limits, but instead per-node limits C_n for all $n \in N$. We must therefore adapt the above definitions accordingly. First and foremost, the definition of the feasibility of a rate vector $r = (r_1, \ldots, r_p)$ needs to be adapted:

DEFINITION 2 A rate vector r is called *feasible* in an overlay setting iff

$$\forall p \in P : \quad r_p \geqslant 0 \quad \text{and}$$

$$\forall n \in N : \quad F_n = \sum_{i \in P_n} r_i \leqslant C_n.$$

That is, where previously the total capacity of each link was not to be exceeded, we now analogously apply such a constraint for the per-node bandwidth limits. With this modification, we can transfer the above definition of a max-min fair rate vector to an overlay network setting based on the modified definition of feasibility.

6.4.3 *Achieving Max-min Fairness*

From a theoretical perspective at least, max-min fairness in the above adapted sense appears to be a very well-suited fairness concept for a system like Tor. The question that now arises is whether such a rate allocation can practically be achieved. Developing a distributed algorithm that dynamically adjusts rates and eventually achieves max-min fairness is not trivial at all. This is confirmed by numerous contributions in this field of research over the past decades, including, e. g., [49, 196]. They all suffer from disadvantages such as additional

overhead caused by many exchanged control messages, high computational costs, inaccuracies, and/or long reaction times and therefore slow adjustment.

An alternative and very elegant way of achieving max-min fairness was presented by Hahne in [97]. It is based on a single round robin scheduler on each outgoing link of each node, which serves all flows equally—i. e., there is one queue per outgoing circuit, and these queues are served round robin if they do contain data. Adapted to the Tor setting, this would analogously mean that one single scheduler serves all circuits equally in a round robin fashion. This alone suffices to achieve max-min fairness.

In order to understand this trait, we need to introduce the notion of a *bottleneck*. The definition that we use here corresponds to the one used in classical max-min fairness, but we again need to adapt it to the specifics of the overlay setting considered here.

DEFINITION 3 For a circuit $p \in P_n$ and a rate vector r, a node n is called a *bottleneck* iff

$$F_n = C_n \quad \text{and} \quad \forall p' \in P_n : r_p \geqslant r_{p'} .$$

Less formally stated, a circuit p has a bottleneck in node n if n's capacity is exhausted and this is the reason that p's rate cannot be increased further without violating the max-min fairness criteria. In [38] it has been proven (for the standard definition of max-min fairness) that a feasible vector r is max-min fair if and only if each circuit p has at least one bottleneck. With very similar means, an analogous theorem can also be proven for our modified definition of max-min fairness in overlays.

THEOREM 1 A feasible vector r is max-min fair iff each circuit p has (related to r) at least one bottleneck.

Proof. Suppose r is max-min fair and $p \in P$ is a circuit without a bottleneck (proof by contradiction). Thus, it follows $\forall n$ with $p \in P_n$ and $F_n = C_n$, $\exists p^* \in P_n$ with $p^* \neq p$ such that $r_{p^*} > r_p$ (cf. Definition 3). Accordingly, the following cases yield a positive δ_n, defined as

$$\delta_n = \begin{cases} C_n - F_n & \text{if } F_n < C_n \\ r_{p^*} - r_p & \text{if } F_n = C_n . \end{cases}$$

Now, increasing r_p by $\min\{ \delta_n \mid \forall n \text{ with } p \in P_n\}$ while simultaneously decreasing r_{p^*} if $F_n = C_n$ by the same value to maintain feasibility of r and without decreasing $r_{p'}$ with $r'_p \leqslant r_p$ results in a contradiction to the max-min fairness property, we initially assumed.

Conversely, suppose each circuit p has a bottleneck. In order to increase any r_p, we have to decrease a $r_{p'}$ of the same bottleneck to maintain feasibility of r. However, since Definition 3 states $F_n = C_n$ and $r'_p \leqslant r_p$, the rate vector r directly suffices our max-min fairness criteria. □

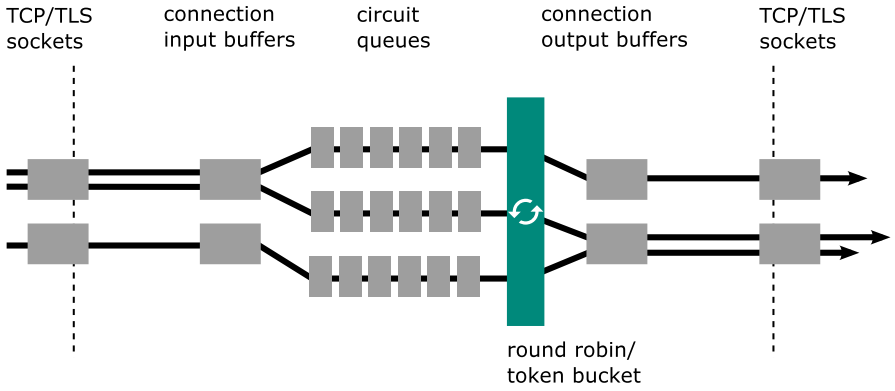


Figure 31: Max-min Fairness scheduling for Tor.

Based on this insight, the emergence of max-min fairness due to appropriate local scheduling can be understood. Consider a circuit whose demand exceeds its allocated rate. If scheduling is performed according to Hahne's proposal, then all cells of this circuit will back up in front of the circuit's bottleneck. This causes a congestion-like situation with a long queue for this circuit. Now the round robin scheduler at this bottleneck node relays cells one-by-one, alternating over all circuits. Because of the backlog of cells, a circuit will never miss an opportunity for sending data through its bottleneck. The result is thus an implicit allocation of a max-min fair share: the allocated rate propagates itself automatically along each circuit and results in a globally max-min fair situation with purely local operations.

Based on these insights, it is possible to re-design the scheduling mechanisms in Tor. In particular, the scheduling mechanisms discussed in the previous section should be replaced by a single rate limiting mechanism (i. e., a token bucket) and a single round robin scheduler on the outgoing side of a relay. Figure 31 illustrates the changes of the scheduling. One can see that only one round robin scheduler remains and iterates over all circuits instead of considering circuits connection-wise only. This scheduler should fairly allocate bandwidth to all circuits which have a non-empty circuit queue and at the same time eliminates the "double door effect".

6.5 EVALUATION

Since many users put their trust in Tor and depend on the services it provides, it became inevitable to test and analyze Tor in safe environments. However, it turned out that for the experimentation with network protocols for Tor under thorough consideration of protocol behavior below the application layer, there is a missing link in the tool chain. Some tools focus only on specific aspects, such as the Tor Path Simulator (TorPS) for circuit paths [111]. Others, such as

Shadow [109] and ExperimenTor [33], run real Tor code, which results in a high degree of realism regarding the application logic, but at the same time requires extensive development efforts to evaluate experimental protocols. While all approaches have their benefits and drawbacks [184], all miss the "noise" of the real Internet and have no real user behavior. Therefore, assumptions about traffic patterns and user behavior are inevitable anyway. The opportunities, though, to vary parameters, scale the setting, and prototype experimental protocols are much more favorable with advanced network simulators such as ns-3 [102].

The code of nstor is publicly available on Github: `https://github.com/tschorsch/nstor`.

Therefore, as a further contribution of this work, we introduce *nstor*, a Tor module for the network simulator ns-3 [102]. It is modeled along the lines of the original Tor software, but clearly focuses on the network aspects. In particular, it includes the transport protocol, the cell transmission scheduler, the token bucket, and the multiplexing. First and foremost, it allows direct and reproducible comparisons of protocol design alternatives in both toy examples and larger scenarios. In addition, with ns-3 the Linux kernel can be hooked, so that practically deployed and widely used transport protocol implementations can be used in the simulations, for additional realism.

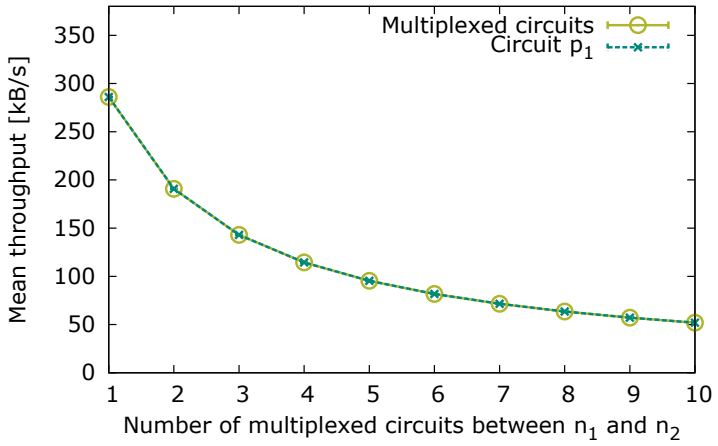
In order to capture the relays' end system nature correctly, we use a star topology where all relays are connected to one central node representing the "Internet". This is a typical setup for the assessment of peer-to-peer overlays. It enforces that all traffic into and out of a relay travels over one single interface. It also allows our simulations to capture effects like, for example, the TCP interferences in overlays that we pointed out in Chapter 3.

Along the lines of the Tor model proposed in [107], we scaled and sampled the Tor consensus (as of 2015/08/04). For realistic circuit paths, we generated a large set of circuits by feeding the scaled consensus to TorPS [111]. As initially argued in Section 6.2, we further assume neither the physically available bandwidth of the end systems nor the Internet backbone are the bottleneck, but that the capacity is bounded by the user-configured maximum allowed bandwidth.

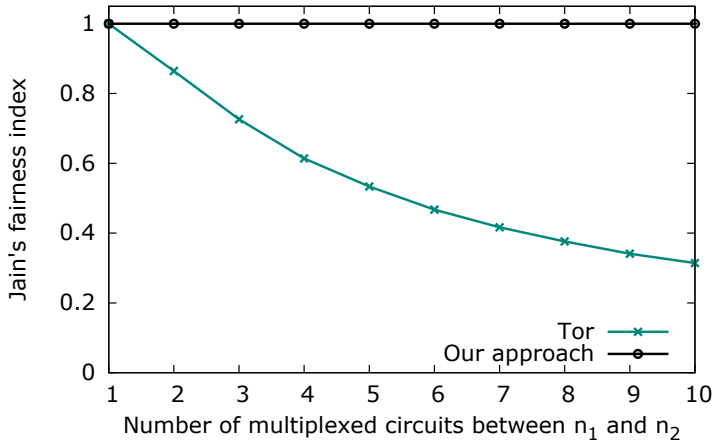
6.5.1 *Fairness in the Simple Test Setting*

Before we turn towards more complex simulation setups, we again take up the scenario used for the illustrative examples in the previous sections. We repeated the same simulations, but now with the here proposed scheduling. Figure 32a shows that, just as expected, the unfairness vanishes: the circuit using one connection alone and the circuits being multiplexed over the other connection all receive equal shares of the available bandwidth.

Figure 32b shows a different perspective on the results. In this figure, we calculated Jain's fairness index [106] for the results with both



(a) Fairness in Tor with max-min fair scheduling.



(b) Jain's fairness index.

Figure 32: Max-min fairness evaluation (simple setting).

scheduling schemes discussed herein ("vanilla" Tor and our approach with the improved scheduler). Jain's fairness index, in our context defined as

$$\frac{(\sum_{p \in P_n} r_p)^2}{|P_n| \cdot \sum_{p \in P_n} {r_p}^2} \quad ,$$

where n is a bottleneck relay, is a measure for the distribution of data rates. It ranges from $1/|P_n|$ as the worst case to 1 as the best case, i. e., uniform distribution. It is important to note, that Jain's fairness index needs to be taken with a grain of salt, because max-min fairness per se does not strive for a uniform distribution. Nevertheless, for $n = n_1$ it clearly underlines the huge improvements: while our proposed mechanism constantly achieves a fairness index very close to 1, the current Tor design exhibits massive unfairness also according to this metric.

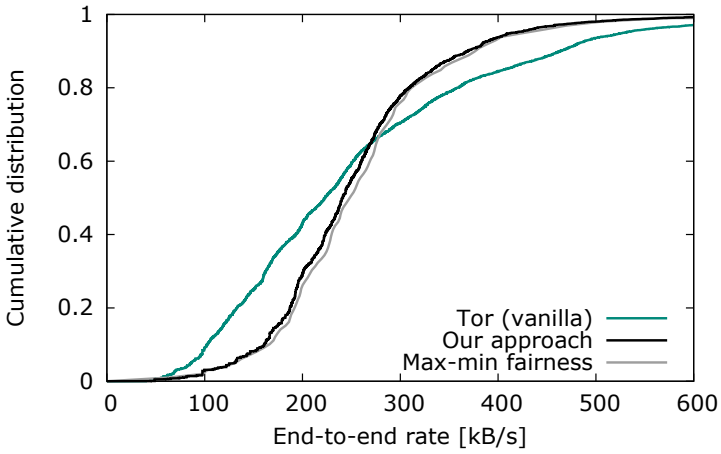


Figure 33: Max-min fairness evaluation (complex setting).

6.5.2 *Fairness in More Complex Settings*

According to the previously presented results, our approach is clearly able to overcome the unfairness problems in the simple setting used above. Now, the question arises how closely we approach the ideal of a globally max-min fair resource allocation in more complex cases.

To answer this question we generated scenarios with 100 relays and 100 circuits according to the Tor consensus as outlined above. We started infinite large downloads (i. e., bulk traffic) over each circuit. After 60 seconds simulation time, we evaluated the steady-state rates, i. e., the mean goodput of the next 60 seconds. For each of the scenarios, we also calculated the perfect max-min fair rate shares offline, with global knowledge. The results of 20 runs are shown in Figure 33 as cumulative distribution with the respective rates on the x-axis. Tor's rate distribution matches daily experiences of Tor users: the performance is rather mediocre with noticeable exceptions in both directions [195]. Our approach, in contrast, allocates rates more evenly distributed with a relative priority to "small" circuits, which is visible from the steeper cumulative distribution function. In fact, it virtually achieves ideal max-min fairness.

In order to quantify how "close" the resulting rate vectors are to the ideal max-min fair rate vector, we calculated the Euclidean distance from the ideal max-min fair vector. The smaller this distance, the closer is the result to a max-min fair rate allocation. In addition, we calculated Jain's fairness index for bottleneck relays. Table 6 shows the quartiles for both metrics. In particular the Euclidean distance makes clearly visible that very large distances are common for Tor. Our approach, in contrast, keeps the distances small, which implies good fairness properties. The results thus confirm our theoretical expectations to the fullest extent and make us believe that a modification of Tor's scheduling as proposed will greatly improve the service.

	Jain's index			Euclid. dist. (kB/s)		
	Q1	Q2	Q3	Q1	Q2	Q3
Tor (vanilla)	0.776	0.833	0.900	1,718	1,783	1,844
Our approach	0.933	0.970	0.991	215	243	295

Table 6: Max-min fairness evaluation.

6.5.3 *Discussion*

We have shown, solving the fairness issues in Tor requires significant effort and consideration, because it involves the question of fairness between circuits taking different paths through the overlay. Compared to this, starvation effects of parallel streams within the same Tor circuit as described in [27], are relatively easy to fix. Both issues, though, show the inefficient resource allocation in Tor.

The gross unfairness between circuits could also be remedied by not multiplexing multiple circuits into one connection, but rather using separate per-circuit connections. If directly applied to Tor's relay architecture, it would resemble our redesigned scheduler by serving all circuits in a round robin manner. For this reason, the authors of TCP-over-DTLS [166] and PCTCP [25]—though, not directly intended—mitigate the fairness problems too. Yet, the severe drawbacks of doing so have been discussed before (cf. Chapter 3).

Our re-designed scheduling, as simple as it is, results in global max-min fairness across the overlay. Additionally it improves the security of anonymity overlays by softening the effect of selective congestion attacks [78, 150], because our scheduler isolates and reduces the impact on concurrent circuits.

However, while having a solution to circuit unfairness at hand, our approach makes another problem in Tor imminent: the above theorem assumes that each circuit sends a huge amount of data, and that a backlog builds up before each circuit's bottleneck. The bandwidth allocation is indeed max-min fair under these assumptions, but the solution alone is not unconditionally practicable and may lead to larger backlogs and thus to increased delays.

In Tor, huge queue sizes are already a problem. As we have shown in the context of the Sniper Attack, queues can virtually grow to any size. The key reason for these problems is Tor' coarse-grained end-to-end sliding window mechanism for congestion control with a fixed window size of 1,000 cells (512 kB). The size is far too much to allow for quick reaction, and it does not provide enough flexibility to adapt to changing situations in the network. With a scheduler as outlined above, the aim must therefore be to keep a non-zero, yet as-small-as-possible number of cells in the circuit queues of the respective bottleneck nodes. We therefore need a mechanism which provides

feedback to upstream nodes along a circuit, to avoid that an excessive amount of cells "pile up", while still maintaining a non-empty queue before the bottleneck relay in order to achieve fairness.

In the following chapter, we will design such a mechanism and build upon the results of this and the previous chapters. In particular, we will incorporate our re-designed scheduler to yield superior fairness.

6.6 CHAPTER SUMMARY

In a first step in this chapter we identified unfairness effects in the Tor anonymity network, caused by the currently employed hierarchical scheduling. With this unfairness in mind we raised the question how to bring local and global fairness to the network. Starting from an existing allocation scheme, we transferred it to overlay networks. Based on this model and the bandwidth constraints in the Tor overlay, we showed how max-min fairness concepts can be applied. In particular, we pointed out that a specific scheduling mechanism can be employed, and that it can indeed achieve global max-min fairness in a network like Tor.

Our approach fits perfectly into the design of the Tor anonymity network, because it operates purely locally and still achieves fairness between circuits on a global scale. Because it does not require explicit coordination and does not need to maintain any explicit state information in order to accomplish this, it also naturally and immediately adapts to any changes in the traffic pattern. In packet level simulations we confirmed the expected improvement and showed the impact on local and global fairness.

In the following chapter, we will continue to pursue our goal of controlling network congestion in anonymity overlays.

7

BACKTAP: BACKPRESSURE-BASED TRANSPORT PROTOCOL

7.1 OVERVIEW

This chapter is based on previous work by the author [9].

So far, we have revealed a number of fundamental issues in the design of Tor, which boil down to a deficient congestion and flow control. We can briefly recap the situation as follows: Tor currently carries all circuits between a consecutive pair of relays in one joint TCP connection. In each relay, cells arriving over one of the TCP connections are demultiplexed, kept in per-circuit queues, and then multiplexed again on the outgoing side, according to the next outgoing connection for the respective circuits. This design is complemented with an end-to-end sliding window mechanism with a fixed, constant window size. Due to its fixed size, this window, quite obviously, lacks adaptivity. As a result, excessive numbers of cells often pile up in the per-circuit queues and/or in the socket buffers of a relay—that is, in the "gap" between the incoming and outgoing TCP connections. Moreover, the large number of inter-relay standard TCP connections results in aggressive aggregate traffic on the one hand, and thus causes unfairness towards other applications in the same network (cf. Chapter 3). And multiplexing varying numbers of circuits into one joint TCP connection is the root of substantial inter-circuit unfairness within Tor on the other hand (cf. Chapter 6).

In this chapter we propose *BackTap: Backpressure-based Transport Protocol*. With BackTap, we take the acquired insights and solutions to design a tailored transport protocol for anonymity overlay. In particular, we replace Tor's end-to-end sliding window by a hop-by-hop backpressure algorithm between relays. Through per-hop flow control on circuit granularity, we allow the upstream node to control its sending behavior according to the variations of the queue size in the downstream node. Semantically, the employed feedback implies "I forwarded a cell". The circuit queue in the downstream relay is therefore, in essence, perceived as nothing but one of the buffers along the (underlay) network path between the outgoing side of the local relay and the *outgoing* side of the next relay. This includes circuit queues and socket buffers into the per-hop congestion control feedback loop, yielding responsiveness and adaptivity. At the same time, it couples the feedback loops of consecutive hops along a circuit, thereby closing the above-mentioned gap. The result is backpressure that propagates along the circuit towards the source if a bottleneck is encountered,

because each local control loop will strive to keep its "own" queue short, while its outflow is directly governed by the next control loop.

We stick to Tor's paradigm of hop-by-hop reliability. However, we implement it in a slightly different way: in the application layer, instead of using TCP's reliability mechanisms between relays. As a result, relays in our architecture have a choice whether to accept or to drop a cell on a per-circuit basis. This avoids reliability-related security flaws as we have discovered in context of the Sniper Attack.

We also do not use a fixed window size, neither end-to-end nor per hop. Instead, we adjust the per-hop window size using an appropriately adapted delay-based congestion control algorithm. In previous applications of delay-based congestion control, first and foremost in TCP Vegas [44], its properties have often been seen as a weakness [15, 17, 46]: it is less aggressive than its loss-based counterparts and therefore tends to be disadvantaged in competitive situations. In our approach, this weakness becomes a strength, because the aggressiveness of aggregate Tor traffic can be a significant problem otherwise.

We implement BackTap including all congestion control mechanisms on the application layer, i. e., in the overlay nodes, based on UDP transport. Consequently, lower-layer changes are not required. A simulation-based evaluation confirms the benefits of the proposed architecture and demonstrates a huge relief of the network regarding congestion.

Our key contributions in this chapter are (a) identifying the "feedback gap" as the primary cause of Tor's performance problems, (b) a novel approach to flow control for environments where data is forwarded over multiple overlay hops, (c) a hop-by-hop backpressure design that avoids network congestion with quick, local adjustments and is therefore well suited to long-delay overlay paths, and (d) an in-depth evaluation with a specific focus on network aspects.

7.2 THE BACKTAP DESIGN

BackTap performs reliability, in-order delivery and flow control on circuit granularity on the application layer. It can be encapsulated in UDP transport, so that there is no need for modifications to the operating system. Communication over a UDP socket also makes the approach scalable to parallel communication with many peers. It also protects from socket exhaustion attacks as described in [90]. In fact, the approach to tunnel tailored transport protocols has become more and more widespread in recent years, the likely best-known example being LEDBAT [180] as used in BitTorrent's µTP [187]. UDP transport can be combined with DTLS [170] or IPsec to provide message integrity and confidentiality, just like Tor currently adds an extra layer of TLS security to its TCP-based overlay links.

In this section, we motivate and present the building blocks of our transport approach in detail. In order to emphasize the changes that we propose and to point out the major design challenges in anonymity networks, we use the current Tor design as a reference architecture throughout the discussion.

7.2.1 *Tor's Feedback Gap*

Since Tor implements another instance of data forwarding and transport functionality on the application layer, overlay mechanisms interact with the behavior of underlay protocols. We have shown these interactions at various stages of this thesis. There are also multiple cases where different mechanisms on both layers have overlapping aims. The prime example is Tor's end-to-end sliding window mechanism: it will obviously interact with TCP congestion control and flow control, which is used between neighboring overlay nodes. This is also at the heart of the feedback gap in the current Tor design, so that the interplay of these two mechanisms is worth revisiting. This will motivate the key design decisions behind our approach.

Recall, Tor relays forward cells according to the circuit switching principle, but the individual relay does not know about the full path of the circuit. Leaving cryptography aside, relays receive cells over TCP, enqueue them to the respective circuit queue and then forward them to the downstream node, again via TCP. The number of cells in flight for any given circuit is limited by the end-to-end sliding window, i. e., 1000 cells, which corresponds to approximately 500 kB. 1000 cells, though, can be significantly more than the bandwidth-delay product of a circuit, which is the reason for long queues and therefore for the huge delays.

Even if the end-to-end window size were not fixed (a possible modification which, of course, has been taken into consideration before [24]), the end-to-end delay of a circuit is too high to dynamically adjust it with reasonable responsiveness. Given the specific situation in anonymity overlays, it is fortunately also not necessary to find an end-to-end solution: because intermediate nodes are aware of individual circuits anyway, relay-supported hop-by-hop feedback with local readjustments based on perceived congestion on the individual overlay hop is a reasonable way out.

What happens, now, if the flow control and congestion control mechanisms of the TCP connections between relays come into play? For the traffic permitted by the end-to-end sliding window, they will determine the local data flow to the next relay. Congestion control will adapt to the underlay network path between the relays. Flow control will specifically depend on the receiving relay's policy for reading from sockets.

This is where the feedback gap appears, which we illustrate in Figure 34a: Tor relays read from incoming TCP connections regardless of the current fill level of corresponding circuit queues in the relay. Therefore, limited outflow of a circuit does not propagate back to the incoming side of the relay. For this reason, the end-to-end sliding window with its non-adaptive constant size and its long feedback loop is the *only* mechanism that limits the number of cells in flight along the circuit, and it is the only mechanism that will eventually throttle the source.

The feedback gap is also where the Sniper Attack appears (cf. Chapter 5). We exploited this design issue to force relays to buffer a huge amount of data until they reach the out-of-memory state. Furthermore, we used the long feedback loop to significantly reduce the attacker's costs. We conclude that the feedback gap is not only the primary reason for poor performance in Tor, but also responsible for a devastating vulnerability.

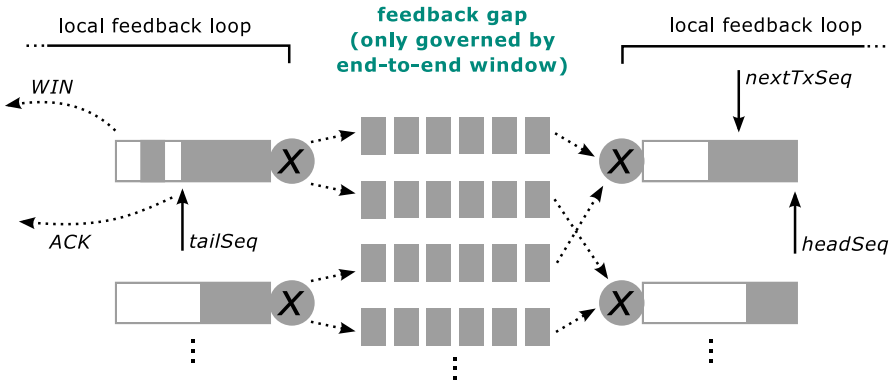
One may then, of course, ask whether it would suffice to stop reading from a circuit's incoming socket if a queue for that circuit builds up locally. This, however, is infeasible because, as discussed before, circuits are multiplexed over joint TCP connections. A relay therefore cannot selectively read cells from one specific circuit; stopping to read from one socket could result in massive head-of-line blocking for other circuits.

Using separate standard, loss-based TCP connections per circuit is also not a good design avenue: as we also discussed before, this would result in excessive numbers of parallel connections, and therefore in very aggressive traffic and high packet loss. In addition, Bufferbloat phenomena [91] cause long reaction times due to excessively large buffers.

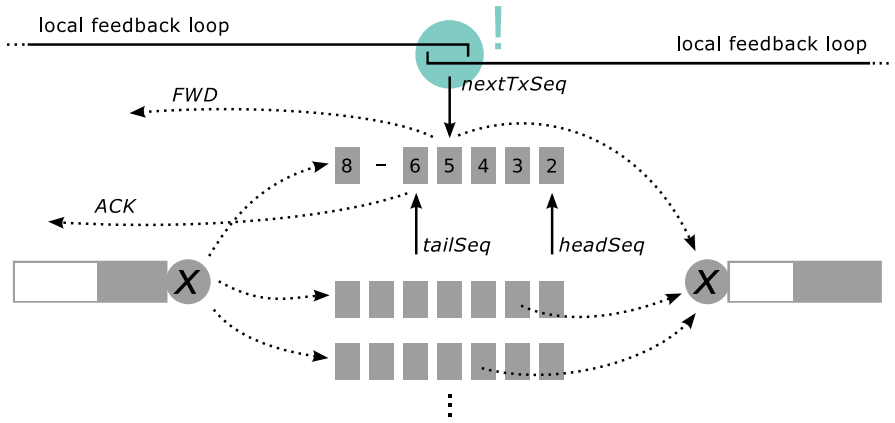
These observations motivate our design based on *delay-based per-circuit congestion control loops*, which can be expected to be much less aggressive than a corresponding loss-based design.

7.2.2 *Realizing Backpressure*

A naive realization of the ideas sketched so far—with separate transport-layer connections per circuit, each with delay-based congestion control—would now likely proceed as follows: if an application-layer queue builds up for one circuit, the inflow might be throttled for that circuit by ceasing to read from its incoming connection. The incoming connection's input buffer would consequently fill up, so that the flow control window is not re-opened; a zero window would be triggered. This would, in turn, throttle the outflow of the upstream node, so that the outgoing socket buffer fills up. The outgoing socket in the upstream node would then no longer be writable, an



(a) Tor's queuing mechanism with cell multiplexing and a feedback gap between ingress and egress, i. e., TCP sockets.



(b) Fused circuit queue that triggers flow control feedback (FWD) not until a cell has been forwarded to the successor to achieve backpressure.

Figure 34: Comparison of feedback loops.

application-layer queue would form there, and so on. Thereby, congestion feedback would propagate indirectly through backpressure.

However, this implies that upstream of the bottleneck overlay link, in each relay there must be enough queued data to fill up (a) the outgoing socket buffer, (b) the application-layer circuit buffer, and (c) the incoming socket buffer. Even keeping technical difficulties related to sizing and management of socket buffers in various operating systems aside, incoming and outgoing socket buffers must at least be sufficiently large to cover the bandwidth-delay product of the respective link, in order not to waste performance. Together with the additional application-layer buffer, the total amount of queued data per overlay hop and circuit would once again have to be very significant, and feedback propagation would likely once again be slow.

To mitigate these effects, we choose a somewhat different, more consequent path: our solution also performs congestion control per circuit, and it likewise does so without multiplexing circuits into joint connections. However, we virtually extend the network into and through the application layer, by emitting flow control feedback only

when a cell has been *forwarded* out of the local relay. The application-layer circuit queues in our design therefore take the role of a fused version of the respective ingress and egress socket buffers. Such a queue is illustrated in Figure 34b, and contrasted with the design that is currently followed in Tor, shown in Figure 34a. The feedback gap in the latter is clearly visible, whereas the local feedback loops in our protocol are directly coupled so that backpressure can build up and propagate immediately upon a deterioration of the available bandwidth.

In the proposed design, arriving cells from the predecessor are read from the UDP socket and processed as usual; that is, in particular the cryptographic operations demanded by the anonymity overlay are performed. The cell is subsequently enqueued in the respective circuit queue. The variable *tailSeq* points to the last cell that has been received in order. *tailSeq* is updated when new cells are received. Cells received out of order may also be queued, with respective gaps in the buffer.

On the other end of the queue, *headSeq* points to the frontmost unacknowledged cell. As soon as we learn that the successor has successfully received the cell, *headSeq* is incremented and the respective cell may be discarded from the buffer.

The third pointer, *nextTxSeq*, is incremented when a cell is forwarded to the downstream relay. The key point that distinguishes our design is: this forwarding at the same time also triggers the transmission of corresponding flow control feedback upstream. We call the respective message an `FWD`. Similar to an `ACK`, an `FWD` also carries a sequence number that refers to a cell. The upstream node can make use of `FWD`s to determine a *sending window* (*swnd*) based on the provided feedback. It is allowed to keep at most *swnd* cells in transmission.

The resulting design is a hybrid between flow control and congestion control: the adjustment strategy to *swnd* follows a delay-based control approach, based on the latency experienced before receiving an `FWD`. It will therefore adjust both to the outflow in the downstream node (because only then the `FWD` feedback is issued) and to the conditions of the overlay network path between consecutive relays (because this network path, too, will influence the delays). In essence, it therefore turns the application-layer circuit buffer into yet another buffer along the network path, without a special role from the perspective of the load feedback.

Moreover, tying `FWD` transmissions to the forwarding of the corresponding cell yields tight feedback coupling between consecutive overlay hops: if the *swnd* adjustment control loop of one overlay hop in a circuit results in a throttled outflow of cells, the `FWD` arrival delay over the preceding overlay hop will increase accordingly within a one-way local-hop delay. *swnd* can therefore be adjusted quickly. This way, hop-by-hop feedback emerges, and backpressure propagates back to

the source. Because delay-based congestion control strives to maintain very short queues, the emerging queues will be small, while available capacity can be fully utilized.

7.2.3 *Reliable Transfer*

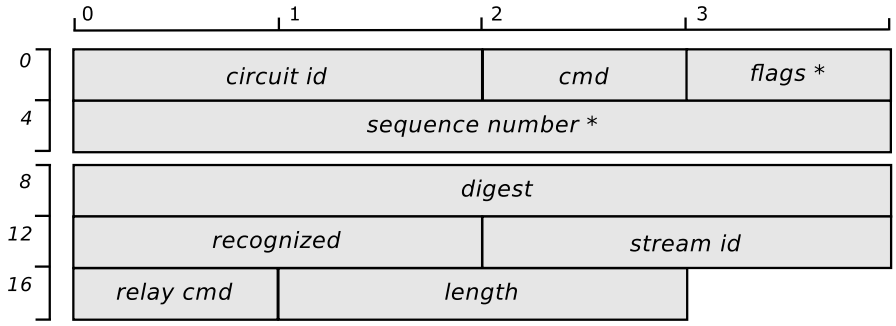
As we know from before, Tor carries application-layer data that expects TCP-like reliable bytestream service. The design relies on each intermediate hop to ensure reliable in-order delivery. That is, there is no end-to-end reliability/acknowledgment/retransmission scheme. Reliability on the individual overlay hop uses the per-hop TCP connections' reliability mechanism; relays are not allowed to drop or reorder cells residing in their per-circuit queues.

We stick to this model also in our proposed transport protocol, i. e., we implement reliability on a per-hop basis. To this end, we use cell sequence numbers to determine the order of cells and to detect losses. The mechanisms generally stick closely to those employed by TCP. The sender infers, either by a timeout or by duplicate acknowledgments, that cells have been lost and retransmits them. The key point where we deviate from TCP's mechanism is where the circuit queue in the downstream node and the coupling between consecutive hop feedback loops comes into play.

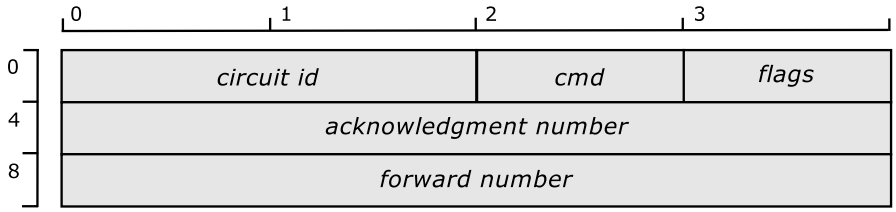
The most consequent version of the philosophy of taking the application-layer circuit queue as "yet another network buffer" would use the `FWD` packets as acknowledgments. This might actually be expected to work reasonably well under many circumstances. However, one could well argue that it potentially wastes resources: after all, when a cell has arrived at the next relay, it is already under the control of the downstream application-layer instance, but reliability feedback is not yet generated. This creates a risk for spurious timeouts, and it might take unnecessarily long to recognize and fix losses.

For this reason, as an optimization, we separate reliability on the one hand and congestion/flow control on the other hand in terms of feedback. We provide reliability feedback as early as possible, namely upon arrival of a cell, by sending a corresponding `ACK`. The calculation of the retransmission timeout (RTO) and the fast retransmit mechanism follow RFCs 6298 [161] and 5681 [21], respectively. Both `ACK`s and `FWD`s are cumulative. Handshakes upon circuit establishment and teardown can likewise closely follow their respective counterparts in TCP.

Implementing reliability on the application layer makes it possible to drop arriving cells by a deliberate decision in the application layer (only before the respective `ACK` has been sent, of course). We note that this opens up new ways out of a difficult problem, i. e., the Sniper Attack, where relays can be attacked by overloading them with cells which they are not allowed to drop. Dropping excessive cells for a



(a) Extended cell header (*new header field).



(b) New feedback cell.

Figure 35: Cell structure.

given circuit is a much cleaner and simpler solution than the heuristics that were necessary to relieve Tor from this threatening attack vector.

In an extended cell structure, we introduce new header fields: a sequence number (4 Byte) and a field for flags (1 Byte). They fulfill comparable roles to the respective fields in the TCP header. However, since cells have a fixed size (for anonymity reasons), sequence numbers refer to cells rater than bytes. The extended cell header is illustrated in Figure 35a.

For `FWD`s and `ACK`s, we introduce a separate message format, much shorter than a Tor cell. In one UDP packet traveling between two relays with a typical MTU, up to two regular cells and, in addition, a number of `FWD`/`ACK` messages—not necessarily for the same circuits—can be encapsulated. The freedom to combine `FWD`s/`ACK`s with cells also from other circuits (or, of course, to send them separately if no cells travel in the opposite direction) corresponds to a generalized variant of piggybacking.

Our modifications to the Tor cell structure affect the cell preamble only. The preamble is not part of the onion encryption and therefore remains unencrypted on the application layer. Likewise, `FWD`/`ACK` messages are not application-layer encrypted. However, note that lower-layer encryption between consecutive relays (TLS or DTLS, respectively) shields the preamble from observers on the wire. In fact, this design provides an additional layer of security by encrypting reliability and flow/congestion control information. When using kernel-level TCP, as Tor does, the respective header fields are not encrypted by TLS.

7.2.4 *Flow Control*

In the proposed protocol design, each sender/forwarder determines the size of its local *swnd* based on the `FWD` feedback from the next hop downstream. `ACK`s are used for reliability, but do not influence the window adjustment.

Most transport protocols, and in particular most TCP variants, use packet loss as an indicator of congestion and therefore as a basis for adjusting their window size or transmission rate; details highly depend on the TCP flavor [15]. The usual suspects of loss-based approaches, i. e., NewReno [101] and CUBIC [95], increase their sending rate to approach the maximum available capacity until packet losses, typically due to queue overflows, occur. Therefore, as we explained in Chapter 3, loss-based TCP can become very aggressive.

Delay-based approaches along the lines of TCP Vegas [44] as they are used here, in contrast, take delay variations rather than packet losses as a signal for congestion. The basic idea behind Vegas is simple: if queuing delays become larger, then decrease the congestion window, else leave it as is or increase the window. If queues, for example, start to build up—that is, *before* losses due to exceeded buffers occur—the measured delay increases and indicate a window size larger than the bandwidth-delay product. The control algorithm re-adjusts the congestion window accordingly. Thus, they are less aggressive in the sense that they do not force losses and do typically not fully utilize buffers in intermediate nodes.

This reduced aggressiveness constitutes a significant benefit for an anonymity overlay. The Tor overlay at this time is formed by more than 6000 relays (with increasing trend [195]) in a fully connected topology. All currently active connections to other relays compete for the available capacity. The resulting traffic, in sum, is very aggressive and inevitably provokes significant packet loss—also for other traffic traversing the same bottleneck. One may expect that this can significantly be reduced by using delay-based controllers.

Following the ideas of TCP Vegas, our aim is to set *swnd* close to the bandwidth-delay product and therefore avoid long queues (ergo achieve short delays). To this end, we calculate the difference between the expected and the actual window size as

$$diff = swnd \cdot \frac{actualRtt - baseRtt}{baseRtt},$$

where *actualRtt* and *baseRtt* are the RTT with and without load. In the literature they are also referred to as the "experienced RTT" and the "real RTT". We sample the RTT based on the flow control feedback by measuring the time difference between sending a cell and receiving the respective `FWD`. The *actualRtt* is estimated by taking the smallest RTT sample during the last RTT. This reduces the effect of outliers

due to jitter on the network path. The *baseRtt* is the minimum over all RTT samples of *all* circuits directed to the *same* relay. The assumption is that occasionally cells will encounter nearly empty queues. Hence, the smallest RTT seen so far is a good estimate of the round-trip propagation delay. Moreover, the individual *diff* calculations per circuit use a joint *baseRtt* estimate. This mitigates potential intra-fairness issues of delay-based approaches.

Depending on the value of *diff*, we adjust the sending window every RTT as follows:

$$swnd' = \begin{cases} swnd + 1 & \text{if } diff < \alpha \\ swnd - 1 & \text{if } diff > \beta \\ swnd & \text{otherwise.} \end{cases}$$

Since *swnd* changes by at most one, it follows an additive increase additive decrease (AIAD) policy. Typically α and β are chosen as 2 and 4 (here measured in cells), respectively. Therefore, one may expect that *swnd* does not exceed the bandwidth-delay product by much. This is sufficient to achieve full utilization of the available capacities.

Combined with our re-designed circuit scheduling algorithm from the previous chapter, which locally round robins all circuits, this adjustment scheme (for the same reasons as before) yields a rate allocation that achieves global max-min fairness between circuits. In addition, prioritization heuristics such as [22, 110, 189] can be applied, if a prioritization of certain traffic types and patterns is desired. End-to-end windows and corresponding feedback are no longer necessary.

7.3 EVALUATION

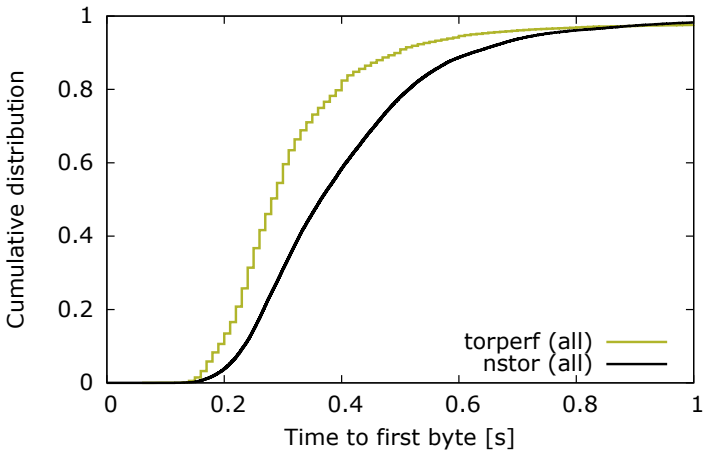
Integrating and evaluating a new transport protocol is a major hurdle by itself [135]. Its true benefits (and potential drawbacks) will only show in a sufficiently large setting with reasonable variability. However, a deployment in a real-world anonymity overlay will only be realistic after very thorough preceding evaluations and in-depth discussion in the research community—a process which we hope to initiate with our work. Even deployments in an emulated or testbed-based anonymity network, are also notoriously hard to analyze—because the anonymity itself prohibits in-depth traceability and measureability. We therefore evaluated the proposed protocol in a large-scale simulation study.

As in the previous chapter, *nstor*, our open source Tor module for the network simulator ns-3, proves to be a valuable tool. It complements the existing tool chain as a simulator particularly suitable for network-focused evaluations of experimental approaches.

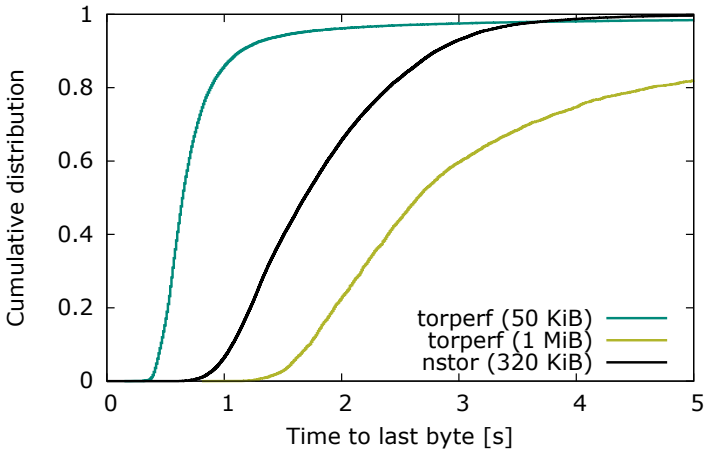
The key point in setting up an environment for a valid evaluation of Tor is to model the overlay appropriately. The Tor model from [107] serves as a guideline here. In our simulations, we use a star topology for simple, easy-to-analyze toy scenarios, and a dumbbell topology for larger-scale, more realistic experiments. Since approximately 93 % of all Tor relays are currently hosted in North America or Europe [195], the dumbbell topology can be thought to approximate the geographical clustering. For this reason, we adjusted the delay according to the iPlane [136] RTT measurements and the client access rates according to Akamai's state of the Internet report [18] by inverse transform sampling, i. e. generating random samples from its cumulative distribution function (CDF). In addition, we scaled and sampled the Tor consensus (as of 2015/08/04) and generated a large set of circuit paths by feeding this consensus to TorPS [111]. Unless otherwise specified, we assumed neither the physically available bandwidth of the relays' access link nor the Internet backbone to be a bottleneck, but that the relay capacity is bounded by the operators using the above-mentioned token bucket rate limiter.

In accordance to the model proposed in [107], we deliberately distinguish only two types of circuits, bulk and web circuits. Bulk circuits continuously transfer 5 MiB files, i. e., after completing such a download they immediately request another one. Web circuits request 320 KiB files with a random "think time" of 1 to 20 seconds between consecutive requests. Although apparently being very simplistic, it is the common approach used by the Tor community and hence increases the comparability to related research. As [107] stresses, the ratio of simulated web and bulk circuits in relation to the number of relays requires calibration to produce network characteristic that approximate Tor. Therefore, we used the publicly available torperf data set [195], which consists of measurements of various file downloads over the live Tor network. The time-to-last-byte (TTLB) and time-to-first-byte (TTFB) results (as of August 2015) are shown in Figure 36 as CDF plots. For our analysis in a larger setting, we observed that a scenario with 100 relays and 375 circuits with 10 % bulk circuits approximates Tor's performance reasonably well (cf. Figure 36). This configuration corresponds to one of Shadow's example scenarios (as of Shadow v1.9.2). In this setting, we simulated a period of 300 seconds (simulation time), started the clients at random times during the first 30 seconds and left the system another 30 seconds lead time before evaluating. For statistically sound results, all simulations in this paper were repeated with varying random seeds and are presented either with 95 % confidence intervals or as cumulative distribution functions.

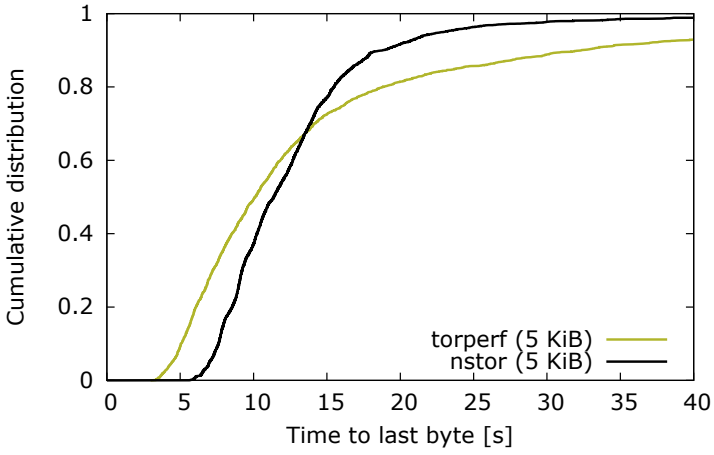
In addition to "vanilla" Tor and our approach, BackTap, we also implemented the N23 protocol as proposed in [24] and PCTCP [25] (which is conceptually identical to TCP-over-DTLS [166]). This con-



(a) Bulk and web circuits.



(b) Web circuits.



(c) Bulk circuits.

Figure 36: Calibration of the simulation environment.

stitutes the first qualitative comparison among alternative transport design proposals for Tor. It also underlines the flexibility of nstor, our ns-3-based simulation module.

7.3.1 *Steady State*

First, we take a look at the steady state behavior, i. e., the point when long-term flows reach equilibrium. For the analysis of these situations, we first focus on the cumulative amount of delivered data of a circuit: by $W(t)$ we denote the amount of payload data delivered to the client up to time t. The counterpart on the sender side is $R(t)$, which denotes the cumulative amount of data injected into a circuit up to time t. Obviously, both functions are non-negative, non-decreasing and $R(t) \geqslant W(t)$ must hold true at all times.

Given R and W, the end-to-end backlog can be defined as $R(t) - W(t)$, the end-to-end delay as $t_2 - t_1$ for $t_1 \leqslant t_2$ and $R(t_1) = W(t_2)$, and the achieved end-to-end data delivery rate during an interval $[t_1, t_2]$ as

$$\frac{W(t_2) - W(t_1)}{t_2 - t_1}.$$

Intuitively, these are the vertical difference, the horizontal difference and the slope of the respective functions. For our simulation, we sampled $R(t)$ and $W(t)$ at the sender side (in Tor often called the "packaging edge") and the receiver side (the "delivering edge") of a circuit every 10 ms (simulation time). After the steady state is reached, we performed a linear regression on our data points and calculated the rate, backlog and delay accordingly. The results for a single circuit with a bottleneck rate of 1 500 kB/s (enforced through an application layer limit at the middle relay) and varying end-to-end RTT are given in Figure 37 as a mean of 20 runs with 95 % confidence intervals.

Since Tor has a fixed window size that it will fully utilize, the results with the standard Tor protocol heavily depend on how this window size relates to the bandwidth-delay product (BDP), and thus to the end-to-end RTT. In our example, the circuit window size matches the BDP at an RTT of approximately

$$1\,000 \cdot \frac{512\,\mathrm{B}}{1\,500\,\mathrm{kB/s}} \approx 341\,\mathrm{ms}.$$

For a smaller BDP, the backlog significant increases the delivery delay; for higher RTTs, the download rate drops and asymptotically converges to zero, because the window does not suffice to fully utilize the available bandwidth. This clearly demonstrates Tor's fundamental problem: on the end-to-end level, the only control mechanism is the fixed window, which, however, does not adapt to the network path.

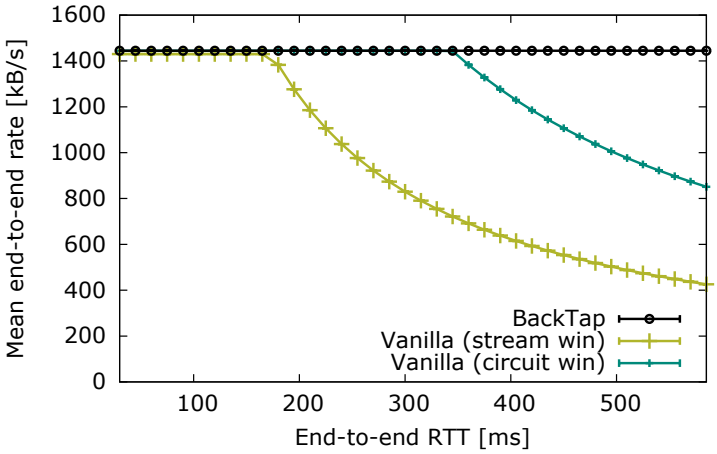
There are some noteworthy phenomena that might be confusing at first sight. In a first approximation according to theory, one would expect that half of a circuit window's worth of data (i. e., approximately 250 kB) is travelling in downstream direction, while the other half of the window is on its way back in the form of SENDME cells. The end-to-end backlog (as defined above: the difference between the amount of sent and received data at a given point in time) should therefore be approximately 250 kB. However, recall that the rate limit is enforced on the application layer by a token bucket. Our model follows the implementation in Tor, where this token bucket is refilled periodically every 100 ms. The bottleneck operates at its capacity limit, always draining its bucket and sending corresponding cell bursts. Thus, about every 100 ms approximately $1\,500\,\text{kB/s} \cdot 100\,\text{ms} = 150\,\text{kB}$ (300 cells) arrive at the client, consequently triggering three SENDMEs. As a result, as long as the RTT is lower than the 100 ms refill interval, only three SENDMEs are on the way back, so that the upstream amount of data is correspondingly higher (about 350 kB). For higher RTTs, the observed backlog approaches the theoretical limit without this effect, i. e., 250 kB. Both levels, 350 kB and 250 kB, can be observed in Figure 37b for vanilla Tor ("circuit win").

The respective end-to-end delay, as seen in Figure 37c, behaves according to the built up backlog. That is, while the circuit window is larger than the BDP, there is a noticeable delay. Ideally, the end-to-end delay should be half the end-to-end RTT, though.

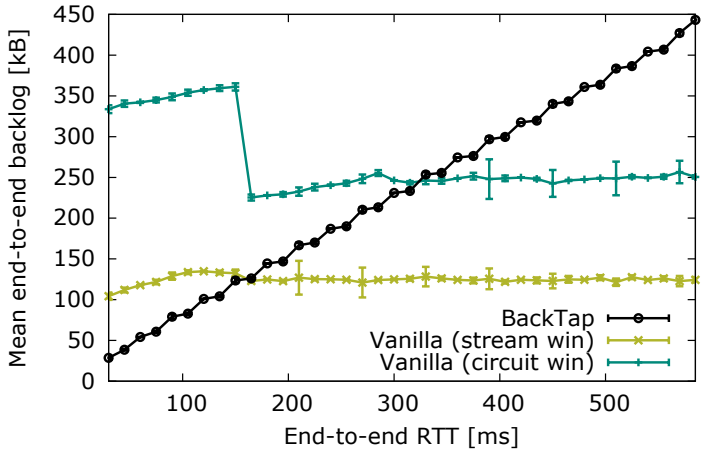
It is important to note that with a fixed window size there is only one sweet spot. If this point is not met, either the backlog and hence the delay increases or the circuit becomes underutilized. A heterogeneous and volatile network such as Tor is condemned to yield poor performance when employing a static mechanism.

Of course, the same applies to simulations where the (smaller) stream window is the limiting factor: the rate drops much earlier, at $500 \cdot 512\,\text{B}/1\,500\,\text{kB/s} \approx 171\,\text{ms}$. While the end-to-end RTT is less than 100 ms, the three SENDMEs in upstream direction cause a backlog of about 100 kB, this time slightly less than half the window size. Beyond this point, the results meet theory and the backlog levels at half the stream window, that is 125 kB.

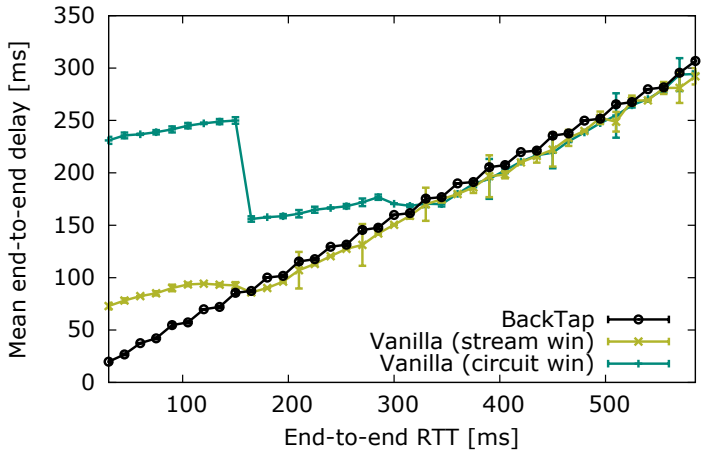
We also observed that Nagle's algorithm [152] can interfere with Tor's window mechanism. In a nutshell, Nagle's algorithm suspends transmission for a short period and tries to combine small chunks of data to reduce the overhead. This behavior causes extra delays upon transmission of SENDMEs, and thereby artificially increases the experienced RTT. As a consequence, the rate drops much earlier and the backlog settles at a lower level accordingly, because a larger fraction of the window is spent on the upstream (SENDME) direction (not shown in the figure). However, as soon as scenarios become more complex including more traffic flows, the effect vanishes. By default Nagle is



(a) End-to-end rate.



(b) End-to-end backlog.



(c) End-to-end delay.

Figure 37: Single circuit scenario clearly demonstrates Tor's fundamental problem and the benefits of our approach.

enabled in today's deployments and hence also in Tor. Therefore, we disabled it only to make the previous simulations more easily comprehensible; in all our following simulations Nagle will be enabled. Nevertheless, either with or without Nagle enabled or with the stream or circuit window in place, a fixed size window is not able to adapt and obviously comes at a severe cost in performance.

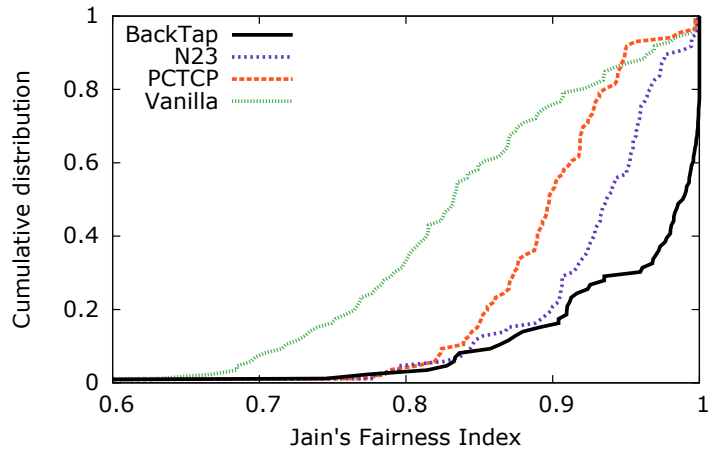
In contrast, our approach is able to adjust to the network in all situations. It maintains the rate, while the backlog increases linearly with the RTT (and thus with the BDP). As a result, we achieve an end-to-end delay that always just slightly exceeds the physical RTT. This is the behavior a good transport protocol should exhibit.

7.3.2 *Fairness*

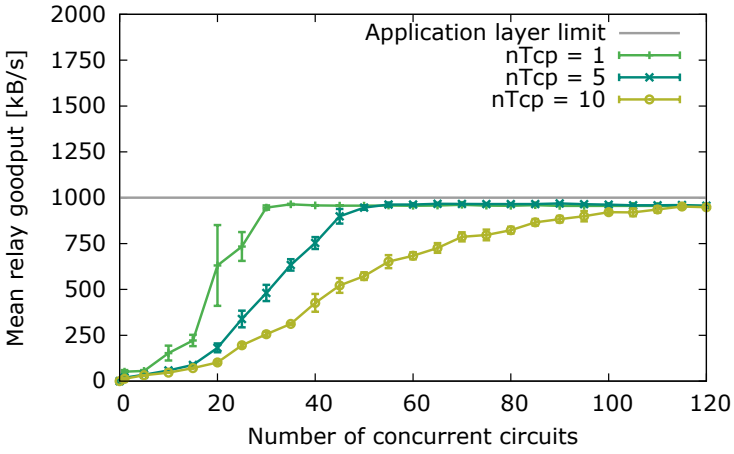
For those readers familiar with delay-based congestion control, a number of typical issues will likely come to mind. In particular, they relate to intra-fairness and inter-fairness. We therefore now assess these aspects.

INTRA-FAIRNESS Delay-based congestion control depends on accurate RTT measurements. In particular, "late coming" connections may suffer from an overestimated *baseRTT*. This leads to intra-fairness issues, i. e., to drawbacks in the competition with other delay-based connections. We mitigate this issue by sharing *baseRTT* information among circuits directed to the same successor. Thus, circuits established at a later point in time will still base their calculations on sound *baseRTT* measurements. This is a feature of our approach that becomes possible, because the transport protocol logic is implemented in the application layer.

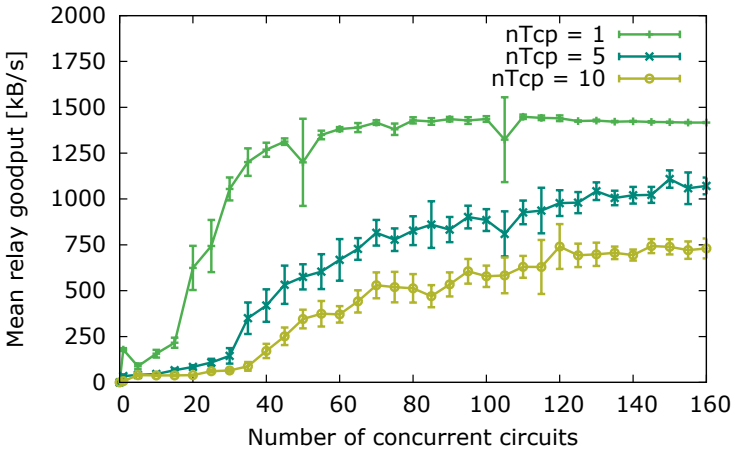
Furthermore, our approach enables cell scheduling on circuit granularity. This avoids the fairness issues due to varying numbers of active circuits multiplexed into one transport layer connection, as described in the previous chapter. Figure 38a shows Jain's fairness index [106] calculated over per-circuit goodputs at the respective bottlenecks. This index quantifies fairness as a value between zero and one, where one means perfect fairness. For this simulation, a star topology with 50 relays and 100 circuits generated according to the real-world Tor consensus were used. We started infinite large downloads (i. e., bulk traffic) over each circuit, where the starting times were randomly distributed during the first 30 seconds. We let the simulation settle for another 60 seconds to reach a steady state before evaluating the mean per-circuit end-to-end rates. The results of 20 runs are given as a cumulative distribution plot. Our approach, in fact, achieves a much fairer distribution than all other protocols, which the larger fraction of higher fairness indices confirms.



(a) Intra-fairness.



(b) Inter-fairness (application layer bottleneck).



(c) Inter-fairness (access link bottleneck).

Figure 38: Fairness evaluation.

	Protocol	$\frac{\text{goodput}}{\text{throughput}}$ ratio	Mean backlog
100 Circs	BackTap	0.89	29 kB
	N23	0.86	112 kB
	PCTCP	0.90	181 kB
	Vanilla	0.90	184 kB

Table 7: Overhead and backlog comparison.

In these simulations, we also investigated the overhead by comparing the ratio of the achieved goodput (on the application layer) and the actually transmitted bytes (on the MAC layer), i. e. the throughput. The results, as seen in Table 7, show an insignificant difference of approximately 1 % compared to vanilla Tor. We also found that our approach largely reduces the number of in-flight cells in the network: the total backlog is about three (in case of N23) to six times (in case of vanilla and PCTCP) lower.

INTER-FAIRNESS One of the most prominent caveats of delay-based approaches is that they are "over-friendly" to concurrent loss-based connections. Basically, they reduce the sending rate before loss-based approaches do, because they detect congestion earlier. In some cases this is an intended behavior (cf. LEDBAT [180]), while in the case of TCP Vegas this was generally perceived as an issue [15, 46]. However, if a number of delay-based sessions come together, they are in sum able to compete well [46]. We exploit the properties of delay-based congestion control, because it allows the anonymity overlay to compete more reasonably with other applications (using loss-based TCP) in the relay operators' networks.

We simulated a scenario with a varying number of parallel circuits (on the x axis) and a likewise varying number of competing loss-based TCP connections (*nTcp*). The TCP connections represent downloads that are performed on the same machine as the Tor relay. In a first setting, we limited the anonymity relay bandwidth to 1 MB/s (again using the token bucket rate limiter), while the access link has twice that capacity. In a second setting, we left Tor virtually unlimited (token bucket configured to 10 MB/s) and let the access link become the bottleneck. The results in Figure 38b and 38c demonstrate that in both settings, relative to the number of TCP connections, for small numbers of circuits the over-friendly behavior of the delay-based controller shows clearly. If a higher number of circuits is active in parallel, they still leave a good fraction of the total 2 MB/s for the competing non-anonymity connections. For typical relays today one may expect between a few hundred and several thousand concurrently open circuits [25]; of course, not all of them are active all the time. We believe that the willingness to yield a substantial part of the bandwidth to other applications constitutes an important incentive for relay operators to donate more bandwidth.

7.3.3 *Larger-Scale Analysis*

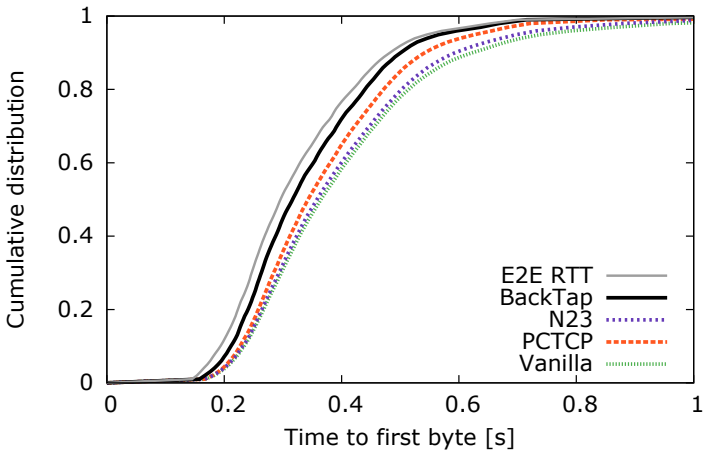
For an analysis in a larger setting, we simulated scenarios with a dumbbell topology and paths generated according to the real-world Tor consensus, as described above. The time-to-first-byte and time-to-last-byte results of the calibrated setting are shown in Figure 39 as CDF plots.

In Figure 39a, we show the TTFB results for web and bulk traffic. Virtually all initial byte sequences of answers to requests are delivered faster with BackTap than with any other protocol. In fact, BackTap's TTFB results are very close to the optimum, i. e., the network's physical end-to-end RTT (denoted as "E2E RTT" in the plot). TTFB is an important measure for the interactivity and has a significant impact on the overall user experience. The lower achieved TTFB would likely result in an increased user satisfaction, due to increased reactivity.

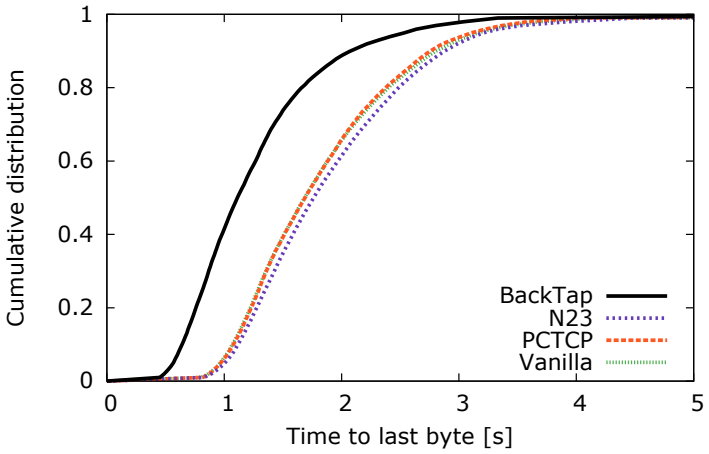
The performance gain of our approach becomes apparent when looking at the TTLB results in Figure 39b and 39c. While the download times for web requests typically vary in the range between 1 and 3 seconds, we achieve a significantly better performance, where almost half of all the requests are already completed in less than 1 second. Also the bulk transfers yield better results, i. e. approximately 30 % more bulk downloads are completed in under 10 seconds.

In order to assess the performance of our approach in a very congested network, we additionally simulated a scenario with 800 circuits. The results are shown in Figure 40. Also in this "stress test" scenario, BackTap is able to achieve reasonable results, which in all cases yield shorter download times. Particularly a look at Figure 40c provides a deeper explanation for these results. It shows that when using our approach, bulk traffic is prevented from "clogging" the network as with the other protocols. As a consequence, we see that bulk downloads take significantly longer to finish. However, this does not mean that bulk traffic is treated unfair: quite in contrast, all of the circuits and flows are treated equally. This is an important feature of our approach: it gives all circuits, web and bulk, a fair share of the network capacity, without the need for (complex, error-prone) explicit traffic pattern analysis and prioritization.

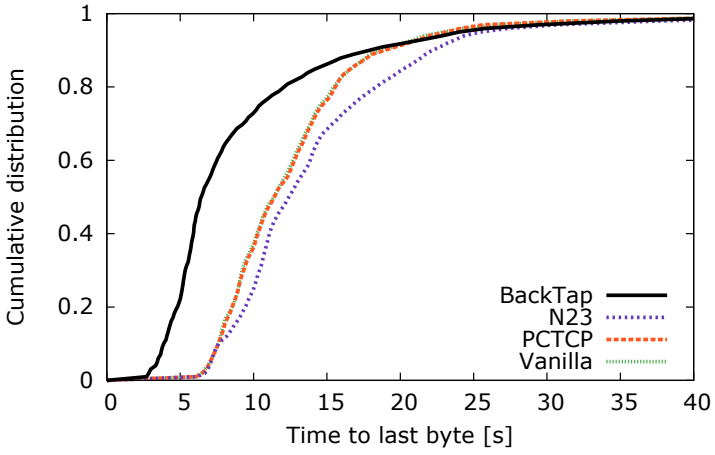
Another perspective on the performance of the various protocols provides Table 8. There we summarize the number of completed downloads and the mean download rate for both larger-scale simulation scenarios. In the stress test with 800 circuits, BackTap is able to complete approximately 5 % and 15 % more web and bulk requests respectively compared to vanilla Tor. Eventually, the mean download rate is in all cases higher as well. On a more general level, we note that vanilla Tor shows, particularly for the web traffic, a much higher variance of TTFB and TTLB. There is, for instance, always a non-negligible fraction of connections that takes far longer than average.



(a) Bulk and web circuits.

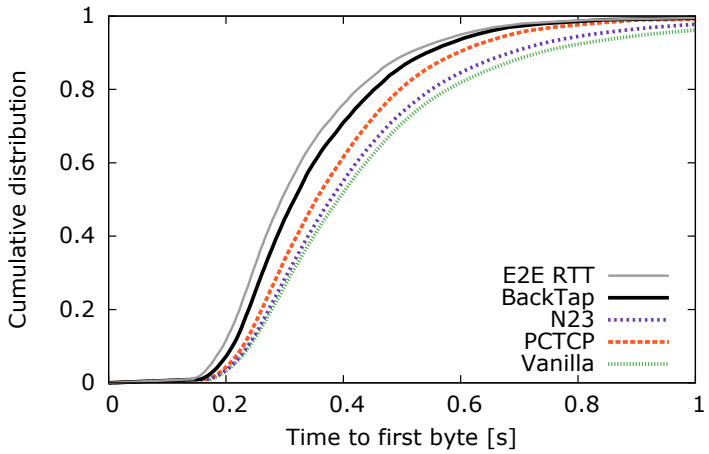


(b) Web circuits.

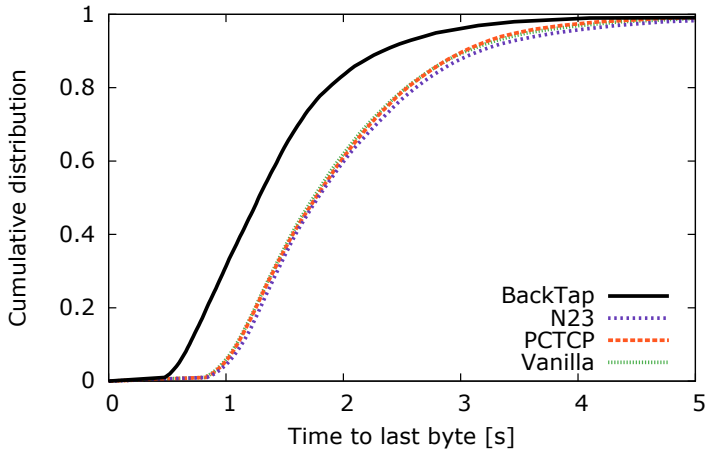


(c) Bulk circuits.

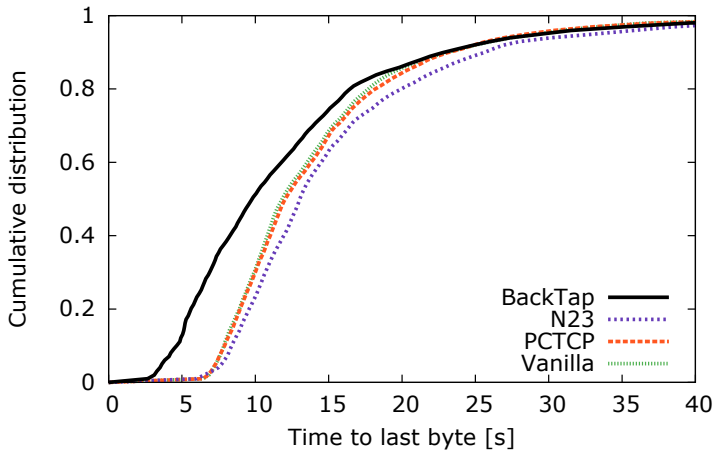
Figure 39: Time to download files over Tor (10 runs, 100 relays, 375 circuits).



(a) Bulk and web circuits.



(b) Web circuits.



(c) Bulk circuits.

Figure 40: Time to download files over Tor (10 runs, 100 relays, 800 circuits).

		Bulk		Web	
	Protocol	#dwnlds	Mean rate	#dwnlds	Mean rate
375 Circs	BackTap	7 503	587 kB/s	52 102	357 kB/s
	N23	4 563	378 kB/s	49 065	215 kB/s
	PCTCP	5 426	424 kB/s	49 513	223 kB/s
	Vanilla	5 493	426 kB/s	49 522	228 kB/s
800 Circs	BackTap	12 108	439 kB/s	110 142	302 kB/s
	N23	9 067	346 kB/s	104 641	204 kB/s
	PCTCP	10 388	376 kB/s	105 288	207 kB/s
	Vanilla	10 491	382 kB/s	105 276	217 kB/s

Table 8: Number of completed downloads (#dwnlds) and mean rate.

This observation is in line with practical experiences of Tor users and the results presented in [107, 195]. Our approach, according to the results presented here, typically reduces the overall variance by more than 17 %.

7.4 CHAPTER SUMMARY

Aware of Tor's fundamental problems and the specific requirements of anonymity overlays, we developed a tailored transport protocol, namely BackTap. In particular, we presented a novel way to couple the local feedback loops for congestion and flow control. It builds upon backpressure between consecutive application-layer relays along a circuit, and a delay-based window size controller. We showed that this can bring a huge relief regarding network congestion by closing the gap between local controllers, so that the need for slow end-to-end control vanishes. In packet level simulations we confirmed the expected improvement.

Besides, there are good reasons why our approach also makes Tor more resilient. First, due to the backpressure, congestion-based attacks will have less influence on other circuits. Second, the much fairer resource allocation makes circuits "look" more "similar", thereby improving the cover traffic properties of concurrent circuits. Third, since BackTap is part of the application layer, transport-related headers are disguised by lower-level encryption as well.

Overall, we believe that our approach shows new ways for designing suitable transport mechanisms for anonymity overlays.

8 CONCLUSION

In this work, we have discussed the design of low-latency anonymous communication systems and developed solutions to a number of challenging questions, all collectively following the aim of enhancing and securing such networks. By doing this we created a fundamental (technical) understanding of networking aspects in anonymity overlays and tackled the most prevalent performance issue experienced today: network congestion.

After discussing the state of the art in anonymous communication in Chapter 2, we systematically explored the design space of transport in anonymity overlays in Chapter 3 and revealed a number of common misconceptions. This led to the conclusion that existing approaches cannot deliver satisfactory performance with any so far considered combination of overlay and underlay protocols. Based on our insights, we paved the ground and derived guidelines of a transport layer design.

In order to be able to assess the efficacy of anonymity overlays, we presented a methodology to measure the network in a privacy-preserving manner in Chapter 4. To this end, we exploited the probabilistic nature of FM sketches. In our analysis, we compared an attacker's a-priori and a-posteriori knowledge and used the relative knowledge gain as a privacy metric. This novel, information-theoretic viewpoint makes design tradeoffs evident and transparent. We believe that the algorithm will be able to support future development and design decisions.

In Chapter 5 we presented the Sniper Attack, a destructive denial of service attack against relays. The attack exposed a fundamental design issue in Tor, that is, performing hop-by-hop reliability and end-to-end flow control implies vulnerabilities. We performed an in-depth security analysis and developed a defense that renders the attack ineffective.

Subsequently, we emphasize the central role of fairness. The notion of fairness has almost entirely been neglected so far in the scientific discussion on anonymity networks, despite being of particular importance. In Chapter 6 we showed that, in the current Tor design, gross unfairness between circuits may arise and lead to poor performance. With this unfairness in mind, we modeled the Tor network and showed how max-min fairness concepts can be applied. In particular, we pointed out that a specific scheduling mechanism can be employed, and that it can indeed yield local and global fairness.

Aware of Tor's fundamental problems and the specific requirements of anonymity overlays, Chapter 7 collects the acquired knowledge to create an advanced solution. In particular we developed BackTap, a tailored transport protocol, which couples the local feedback loops for congestion and flow control in a novel way. It builds upon backpressure between consecutive relays along a circuit, and a delay-based window size adjustments. We showed that this can bring a huge relief regarding network congestion by closing the feedback gap between local controllers.

In summary, each chapter contributed new insights and solution strategies, which influenced our tailored transport protocol. But likewise, each individual aspect provided a contribution, which is able to improve the network. Most of the addressed issues were previously unknown. For those pointed out before, the pre-existing approaches more often than not focused on isolated symptoms instead of pushing forward to the root cause. Based on our research, we have ascertained our thesis statement, that is, an integral perspective on networking aspects, as provided by this thesis, is inevitable to tackle the root cause of the prevalent performance issues.

In addition to the substantial performance benefits that can be obtained by using our proposed techniques, our work also helps strengthening the anonymity. Either implicitly by attracting new users, which in sum increase the anonymity set, or explicitly by tuning the quality of the cover traffic and impeding congestion attacks. At all times, the security and privacy aspects of anonymity overlays guided our design decisions.

In the future, the tradeoff of anonymity and performance needs further investigation, though. Additionally, scalability will continue to be a pressing issue, because resources remain scarce. For the latter, this thesis already provides first steps and approaches: BackTap with its dynamic mechanisms to handle heavy load and to resolve congestion will help to sustain a larger user base. Since it is based on UDP, it will also scale to a larger network. Furthermore, our privacy-preserving measurement framework can be used to assist in making future development decisions by providing the dimension of the network and many other metrics.

To conclude, we are convinced that this thesis provides the foundation of networking aspects in anonymity overlays in general and the future development of transport protocols in such networks in particular. We are delighted to see that some of our approaches already made an impact on the scientific discussion [22, 25, 26, 77, 90, 94] and have seen practical considerations [4, 12, 66, 131, 133, 141].

BIBLIOGRAPHY

PUBLICATIONS BY THE AUTHOR

[1] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann. "The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network." In: *NDSS '14: Proceedings of the Network and Distributed System Security Symposium*. San Diego, CA, USA, Feb. 2014.

[2] D. Marks, F. Tschorsch, and B. Scheuermann. "Unleashing Tor, BitTorrent & Co.: How to Relieve TCP Deficiencies in Overlays." In: *LCN '10: Proceedings of the 35th Annual IEEE International Conference on Local Computer Networks*. Denver, CO, USA, Oct. 2010, pp. 320–323.

[3] D. Marks, F. Tschorsch, and B. Scheuermann. *Unleashing Tor, BitTorrent & Co.: How to Relieve TCP Deficiencies in Overlays (Extended Version)*. Tech. rep. TR-2010-001. Computer Science Department, Heinrich Heine University, Düsseldorf, Germany, Aug. 2010.

[4] F. Tschorsch and B. Scheuermann. *Tor Proposal 182: Credit Bucket*. June 2011. URL: `https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/182-creditbucket.txt`.

[5] F. Tschorsch and B. Scheuermann. "An Algorithm for Privacy-Preserving Distributed User Statistics." In: *Elsevier Computer Networks* 57 (14 Oct. 2013), 2775–2787.

[6] F. Tschorsch and B. Scheuermann. "Distributed Privacy-Aware User Counting." In: *HotPETs '11: 4th Workshop on Hot Topics in Privacy Enhancing Technologies*. Waterloo, Canada, July 2011.

[7] F. Tschorsch and B. Scheuermann. "How (not) to Build a Transport Layer for Anonymity Overlays." In: *PADE '12: Proceedings of the ACM Sigmetrics/Performance Workshop on Privacy and Anonymity for the Digital Economy*. London, UK, June 2012.

[8] F. Tschorsch and B. Scheuermann. "How (not) to Build a Transport Layer for Anonymity Overlays." In: *ACM SIGMETRICS Performance Evaluation Review* 40 (4 Mar. 2013), pp. 101–106.

[9] F. Tschorsch and B. Scheuermann. "Mind the Gap: Towards a Backpressure-Based Transport Protocol for the Tor Network." In: *NSDI '16: Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation*. Santa Clara, CA, USA, Mar. 2016.

[10] F. Tschorsch and B. Scheuermann. *Tor is unfair – And what to do about it*. Tech. rep. 481. Computer Science Department, University of Würzburg, Germany, May 2011.

[11] F. Tschorsch and B. Scheuermann. "Tor is unfair – And what to do about it." In: *LCN '11: Proceedings of the 36th Annual IEEE International Conference on Local Computer Networks*. Bonn, Germany, Oct. 2011, pp. 432–440.

[12] F. Tschorsch and B. Scheuermann. *Tor Proposal 183: Refill Intervals*. Dec. 2010. URL: https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/183-refillintervals.txt.

OTHER PUBLICATIONS

[13] C. Abdelberi, P. Manils, and M. A. Kâafar. "Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network." In: *NSS '10: Proceedings of the 4th International Conference on Network and System Security*. Melbourne, Australia, Sept. 2010, pp. 167–174.

[14] F. Adamsky, S. A. Khayam, R. Jager, and M. Rajarajan. "Security Analysis of the Micro Transport Protocol with a Misbehaving Receiver." In: *CyberC '12: Proceedings of the 4th International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. Sanya, China, Oct. 2012.

[15] A. Afanasyev, N. Tilley, P. L. Reiher, and L. Kleinrock. "Host-to-Host Congestion Control for TCP." In: *IEEE Communications Surveys and Tutorials* 12 (3) (2010), pp. 304–342.

[16] R. Agrawal and R. Srikant. "Privacy-Preserving Data Mining." In: *SIGMOD '00: Proceedings of the ACM SIGMOD International Conference on Management of Data*. Dallas, TX, USA, May 2000, pp. 439–450.

[17] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan. "Evaluation of TCP Vegas: Emulation and Experiment." In: *SIGCOMM '95: Proceedings of the 1995 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. Cambridge, MA, USA, Aug. 1995, pp. 185–205.

[18] Akamai. *State of the Internet Report*. Q1 2015.

[19] M. Akhoondi, C. Yu, and H. V. Madhyastha. "LASTor: A Low-Latency AS-Aware Tor Client." In: *SP '12: Proceedings of the 33th IEEE Symposium on Security and Privacy*. San Francisco, CA, USA, May 2012.

[20] S. Akhshabi and C. Dovrolis. "The evolution of layered protocol stacks leads to an hourglass-shaped architecture." In: *SIGCOMM '11: Proceedings of the 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. Toronto, ON, Canada, Aug. 2011, pp. 206–217.

[21] M. Allman, V. Paxson, and E. Blanton. *TCP Congestion Control*. RFC 5681 (Draft Standard). Internet Engineering Task Force, Sept. 2009.

[22] M. AlSabah, K. Bauer, and I. Goldberg. "Enhancing Tor's Performance using Real-time Traffic Classification." In: *CCS '12: Proceedings of the 19th ACM Conference on Computer and Communications Security*. Raleigh, NC, USA, Oct. 2012, pp. 73–84.

[23] M. AlSabah, K. S. Bauer, T. Elahi, and I. Goldberg. "The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting." In: *PETS '13: Proceedings of the 13th Workshop on Privacy Enhancing Technologies*. Bloomington, Indiana, USA, July 2013, pp. 143–163.

[24] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker. "DefenestraTor: Throwing out Windows in Tor." In: *PETS '11: Proceedings of the 11th Privacy Enhancing Technologies Symposium*. Waterloo, ON, Canada, July 2011.

[25] M. AlSabah and I. Goldberg. "PCTCP: per-circuit TCP-over-IPsec transport for anonymous communication overlay networks." In: *CCS '13: Proceedings of the 20th ACM Conference on Computer and Communications Security*. Berlin, Germany, Oct. 2013, pp. 349–360.

[26] M. AlSabah and I. Goldberg. *Performance and Security Improvements for Tor: A Survey*. Cryptology ePrint Archive, Report 2015/235. 2015.

[27] M. Alsabah and I. Goldberg. *Stream starvation in Tor*. or-dev mailing list. Nov. 2010. URL: http://archives.seul.org/or/dev/Nov-2010/msg00039.html.

[28] Y. Amir, B. Awerbuch, C. Danilov, and J. Stanton. "Global Flow Control for Wide Area Overlay Networks: A Cost-Benefit Approach." In: *OPENARCH '02: Proceedings of the 5th IEEE Conference on Open Architectures and Network Programming*. New York City, NY, USA, June 2002, pp. 155–166.

[29] Y. Amir and C. Danilov. "Reliable Communication in Overlay Networks." In: *DSN '03: Proceedings of the 33rd International Conference on Dependable Systems and Networks*. Lisbon, Portugal, June 2003, pp. 511–520.

[30] F. Baccelli, A. Chaintreau, Z. Liu, A. Riabov, and S. Sahu. "Scalability of Reliable Group Communication Using Overlays." In: *INFOCOM '04: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*. Hong Kong, China, Mar. 2004.

[31] H. Balakrishnan, V. Padmanabhan, G. Fairhurst, and M. Sooriyabandara. *TCP Performance Implications of Network Path Asymmetry*. RFC 3449 (Best Current Practice). Internet Engineering Task Force, Dec. 2002.

[32] J. Ball, B. Schneier, and G. Greenwald. *NSA and GCHQ target Tor network that protects anonymity of web users*. The Guardian. Oct. 2013. URL: http://www.theguardian.com/world/2013/oct/04/nsa-gchq-attack-tor-network-encryption.

[33] K. S. Bauer, M. Sherr, and D. Grunwald. "ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation." In: *CSET '11: Proceedings of the 4th Workshop on Cyber Security Experimentation and Test*. San Francisco, CA, USA, Aug. 2011.

[34] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. "Low-Resource Routing Attacks Against Tor." In: *WPES '07: Proceedings of the ACM Workshop on Privacy in the Electronic Society*. Alexandria, VA, USA, Oct. 2007, pp. 11–20.

[35] T. Bayes. "An essay towards solving a problem in the doctrine of chances." In: *Phil. Trans. of the Royal Soc. of London* 53 (1763), pp. 370–418.

[36] S. M. Bellovin and W. R. Cheswick. *Privacy-enhanced searches using encrypted Bloom filters*. Tech. rep. CUCS-034-07. Department of Computer Science, Columbia University, 2007.

[37] O. Berthold, H. Federrath, and S. Köpsell. "Web MIXes: A system for anonymous and unobservable Internet access." In: *PET '00: Proceedings of the International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. Berkeley, CA, USA, July 2000.

[38] D. Bertsekas and R. Gallager. *Data Networks*. 2nd ed. Prentice Hall, Jan. 1992. ISBN: 0132009161.

[39] B. H. Bloom. "Space/Time Trade-offs in Hash Coding with Allowable Errors." In: *Communications of the ACM* 13 (7) (1970), pp. 422–426.

[40] B. H. Bloom. "Space/time trade-offs in hash coding with allowable errors." In: *Communications of the ACM* 13 (7) (1970), pp. 422–426.

[41] M. S. Blumenthal and D. D. Clark. "Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world." In: *ACM Transactions Internet Technology (TOIT)* 1 (1) (2001), pp. 70–109.

[42] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. *Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*. RFC 3135 (Informational). Internet Engineering Task Force, June 2001.

[43] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. "Denial of Service or Denial of Security?" In: *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*. Alexandria, VA, USA, Oct. 2007.

[44] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. "TCP Vegas: New Techniques for Congestion Detection and Avoidance." In: *SIGCOMM '94: Proceedings of the 1994 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. London, UK, Aug. 1994, pp. 24–35.

[45] T. Brenann. *OWASP HTTP Post Tool*. https://www.owasp.org/index.php/OWASP_HTTP_Post_Tool.

[46] L. Budzisz, R. Stanojevic, A. Schlote, F. Baker, and R. Shorten. "On the Fair Coexistence of Loss- and Delay-Based TCP." In: *IEEE/ACM Transactions Networking* 19 (6) (2011), pp. 1811–1824.

[47] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. "SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics." In: *USENIX Security '10: Proceedings of the 19th USENIX Security Symposium*. Washington, DC, USA, Aug. 2010.

[48] J. Camenisch and A. Lysyanskaya. "A Formal Treatment of Onion Routing." In: *CRYPTO '05: Proceedings of the 25th Annual International Cryptology Conference*. Santa Barbara, CA, USA, Aug. 2005, pp. 169–187.

[49] Z. Cao and E. W. Zegura. "Utility Max-Min: An Application-Oriented Bandwidth Allocation Scheme." In: *INFOCOM '99: Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies*. New York, NY, USA, Mar. 1999, pp. 793–801.

[50] S. Chakravarty, G. Portokalidis, M. Polychronakis, and A. D. Keromytis. "Detecting Traffic Snooping in Tor Using Decoys." In: *RAID '11: Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection*. Menlo Park, CA, USA, Sept. 2011, pp. 222–241.

[51] D. L. Chaum. "Untraceable Electronic Mail, Return addresses, and Digital Pseudonyms." In: *Communications of the ACM* 24 (2) (Feb. 1981).

[52] D. Chaum, F. Javani, A. Kate, A. Krasnova, J. de Ruiter, and A. T. Sherman. *cMix: Anonymization by High-Performance Scalable Mixing*. Tech. rep. 2016/008. IACR Cryptology ePrint Archive, 2016.

[53] B. Chen, K. LeFevre, and R. Ramakrishnan. "Privacy skyline: Privacy with multidimensional adversarial knowledge." In: *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*. Vienna, Austria, Sept. 2007, pp. 770–781.

[54] D. Clark, B. Lehr, S. Bauer, P. Faratin, R. Sami, and J. Wroclawsk. "Overlay Networks and the Future of the Internet." In: *Communication & Strategies* 63 (2006).

[55] D. D. Clark. "The design philosophy of the DARPA internet protocols." In: *SIGCOMM '88: Proceedings of the 1988 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. Stanford, CA, USA, Aug. 1988, pp. 106–114.

[56] B. Cohen. *BEP 3: The Bittorrent Protocol Specification*. Bittorrent.org, Jan. 2008. URL: http://bittorrent.org/beps/bep%5C_0003.html.

[57] J. Considine, F. Li, G. Kollios, and J. Byers. "Approximate Aggregation Techniques for Sensor Databases." In: *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*. Boston, MA, USA, Mar. 2004, pp. 449–460.

[58] E. D. Cristofaro, P. Gasti, and G. Tsudik. *Fast and Private Computation of Cardinality of Set Intersection and Union*. Cryptology ePrint Archive, Report 2011/141. 2011. URL: http://eprint.iacr.org.

[59] G. Danezis. *An anomaly-based censorship-detection system for Tor*. Tech. rep. 2011-09-001. The Tor Project, Sept. 2011. URL: https://research.torproject.org/techreports/detector-2011-09-09.pdf.

[60] G. Danezis and R. Clayton. "Introducing Traffic Analysis." In: *Digital privacy: Theory, Technologies, and Practices*. Ed. by A. Acquisti, S. Gritzalis, C. Lambrinoudakis, and S. di Vimercati. CRC Press, 2007. Chap. 5, pp. 95–112.

[61] G. Danezis, R. Dingledine, and N. Mathewson. "Mixminion: Design of a Type III Anonymous Remailer Protocol." In: *SP '03: Proceedings of the 24th IEEE Symposium on Security and Privacy*. Oakland, CA, USA, May 2003, pp. 2–15.

[62] N. Danner, S. Defabbia-Kane, D. Krizanc, and M. Liberatore. "Effectiveness and Detection of Denial-of-Service Attacks in Tor." In: *ACM TISSEC* 15 (3) (Nov. 2012). ISSN: 1094-9224. DOI: 10.1145/2382448.2382449.

[63] A. Das and N. Borisov. "Securing Anonymous Communication Channels under the Selective DoS Attack." In: *FC '13: Proceedings of the 17th International Conference on Financial Cryptography and Data Security*. Okinawa, Japan, Apr. 2013.

[64] P. Dhungel, M. Steiner, I. Rimac, V. Hilt, and K. Ross. "Waiting for Anonymity: Understanding Delays in the Tor Overlay." In: *P2P '10: Proceedings of the 10th IEEE International Conference on Peer-to-Peer Computing*. Delft, The Netherlands, Aug. 2010.

[65] C. Diaz, S. Seys, J. Claessens, and B. Preneel. "Towards measuring anonymity." In: *PET '02: Proceedings of Privacy Enhancing Technologies Workshop*. San Francisco, CA, USA, Apr. 2002, pp. 184–188.

[66] R. Dingledine. *#6252 didn't go far enough*. June 2013. URL: https://trac.torproject.org/projects/tor/ticket/9063.

[67] R. Dingledine. *#9063 enables Guard discovery in about an hour by websites*. June 2013. URL: https://trac.torproject.org/projects/tor/ticket/9072.

[68] R. Dingledine. *Compare performance of TokenBucketRefillInterval params in simulated network*. Sept. 2011. URL: https://trac.torproject.org/projects/tor/ticket/4086.

[69] R. Dingledine, N. Hopper, G. Kadianakis, and N. Mathewson. "One fast guard for life (or 9 months)." In: *HotPETs '14: 7th Workshop on Hot Topics in Privacy Enhancing Technologies*. Amsterdam, Netherlands, July 2014.

[70] R. Dingledine and N. Mathewson. "Anonymity Loves Company: Usability and the Network Effect." In: *WEIS '06: Proceedings of the 5th Workshop on the Economics of Information Security*. Cambridge, UK, June 2006.

[71] R. Dingledine and N. Mathewson. *Tor Protocol Specification*. URL: https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt.

[72] R. Dingledine, N. Mathewson, and P. Syverson. "Tor: The Second-Generation Onion Router." In: *USENIX Security '04: Proceedings of the 13th USENIX Security Symposium*. San Diego, CA, USA, Aug. 2004, pp. 303–320.

[73] R. Dingledine and S. J. Murdoch. *Performance Improvements on Tor or, Why Tor is slow and what we're going to do about it*. Mar. 2009. URL: www.torproject.org/press/presskit/2009-03-11-performance.pdf.

[74] W. Du and Z. Zhan. "Using randomized response techniques for privacy-preserving data mining." In: *KDD '03*. Washington, D.C., 2003, pp. 505–510.

[75] W. Eddy. *TCP SYN Flooding Attacks and Common Mitigations*. RFC 4987 (Informational). Internet Engineering Task Force, Aug. 2007.

[76] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg. "Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor." In: *WPES '12: Proceedings of the ACM Workshop on Privacy in the Electronic Society*. Raleigh, NC, USA, Oct. 2012.

[77] E. Erdin, C. Zachor, and M. H. Gunes. "How to Find Hidden Users: A Survey of Attacks on Anonymity Networks." In: *IEEE Communications Surveys and Tutorials* 17 (4) (2015), pp. 2296–2316. DOI: 10.1109/COMST.2015.2453434. URL: http://dx.doi.org/10.1109/COMST.2015.2453434.

[78] N. S. Evans, R. Dingledine, and C. Grothoff. "A practical congestion attack on tor using long paths." In: *USENIX Security '09: Proceedings of the 18th USENIX Security Symposium*. Montreal, Canada, Aug. 2009, pp. 33–50.

[79] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon. "Prefix-Preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme." In: *Elsevier Computer Networks*. Vol. 46. 2. Oct. 2004, pp. 253–272.

[80] J. Feigenbaum, A. Johnson, and P. F. Syverson. "A Model of Onion Routing with Provable Anonymity." In: *FC '07: Proceedings of the 11th International Conference on Financial Cryptography and Data Security*. Scarborough, Trinidad/Tobago, Feb. 2007, pp. 57–71.

[81] P. Flajolet and G. N. Martin. "Probabilistic counting algorithms for data base applications." In: *Journal of Computer and System Sciences* 31 (2) (Oct. 1985), pp. 182–209.

[82] S. Floyd, M. Handley, J. Padhye, and J. Widmer. *TCP Friendly Rate Control (TFRC): Protocol Specification*. RFC 5348 (Proposed Standard). Internet Engineering Task Force, Sept. 2008.

[83] S. Floyd and K. Fall. "Promoting the use of end-to-end congestion control in the Internet." In: *IEEE/ACM Trans. Netw.* 7 (4) (Aug. 1999).

[84] S. Floyd and V. Jacobson. "Random early detection gateways for congestion avoidance." In: *IEEE/ACM Transactions on Networking* 1 (4) (1993), pp. 397–413.

[85] B. Ford. "Structured Streams: a New Transport Abstraction." In: *SIGCOMM '07: Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. Kyoto, Japan, Aug. 2007.

[86] M. J. Freedman and R. Morris. "Tarzan: A Peer-to-Peer Anonymizing Network Layer." In: *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*. Washington, DC, USA, Nov. 2002, pp. 193–206.

[87] K. Frikken. "Privacy-preserving set union." In: *Applied Cryptography and Network Security*. June 2007, pp. 237–252.

[88] X. Fu, W. Yu, S. Jiang, S. Graham, and Y. Guan. "TCP Performance in Flow-Based Mix Networks: Modeling and Analysis." In: *IEEE Trans. Parallel Distrib. Syst.* 20 (5) (May 2009).

[89] J. Geddes, R. Jansen, and N. Hopper. "How Low Can You Go: Balancing Performance with Anonymity in Tor." In: *PETS '13: Proceedings of the 13th Workshop on Privacy Enhancing Technologies*. Bloomington, Indiana, USA, July 2013.

[90] J. Geddes, R. Jansen, and N. Hopper. "IMUX: Managing Tor Connections from Two to Infinity, and Beyond." In: *WPES '14: Proceedings of the 13th ACM Workshop on Privacy in the Electronic Society*. Scottsdale, AZ, USA, Nov. 2014, pp. 181–190.

[91] J. Gettys and K. Nichols. "Bufferbloat: Dark Buffers in the Internet." In: *Queue* 9 (11) (Nov. 2011).

[92] I. Goldberg, D. Stebila, and B. Ustaoglu. "Anonymity and one-way authentication in key exchange protocols." In: *Designs, Codes and Cryptography* 67 (2) (2013), pp. 245–269.

[93] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. "Hiding Routing Information." In: *IHW '01: Proceedings of the 1st International Workshop on Information Hiding*. Cambridge, U.K., May 1996.

[94] D. Gopal and N. Heninger. "Torchestra: reducing interactive traffic delays over tor." In: *WPES '12: Proceedings of the ACM Workshop on Privacy in the Electronic Society*. Raleigh, NC, USA, Oct. 2012, pp. 31–42.

[95] S. Ha, I. Rhee, and L. Xu. "CUBIC: a new TCP-friendly high-speed TCP variant." In: *ACM SIGOPS Operating Systems Review* 42 (5) (2008), pp. 64–74.

[96] S. Hahn and K. Loesing. *Privacy-preserving Ways to Estimate the Number of Tor Users*. Tech. rep. 2010-11-001. The Tor Project, Nov. 2010. URL: https://research.torproject.org/techreports/countingusers-2010-11-30.pdf.

[97] E. L. Hahne. "Round-Robin Scheduling for Max-Min Fairness in Data Networks." In: *IEEE Journal on Selected Areas in Communications (JSAC)* 9 (7) (1991), pp. 1024–1039.

[98] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk. *QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2*. IETF Internet Draft. 2016.

[99] C. Hazay and K. Nissim. "Efficient set operations in the presence of malicious adversaries." In: *Public Key Cryptography – PKC 2010* 6056 (2010), pp. 312–331.

[100] Q. He and M. Ammar. "Congestion Control and Message Loss in Gnutella Networks." In: *MMCN '04: Proceedings of Multimedia Computing and Networking*. San Jose, CA, USA, Jan. 2004.

[101] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida. *The NewReno Modification to TCP's Fast Recovery Algorithm*. RFC 6582 (Proposed Standard). Internet Engineering Task Force, Apr. 2012.

[102] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena. "Network simulations with the ns-3 simulator." In: *SIGCOMM '08: Proceedings of the 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. Seattle, USA, Aug. 2008.

[103] B. Hubert. *The Wonder Shaper*. URL: http://lartc.org/wondershaper/.

[104] *I2P: The Invisible Internet Project*. URL: https://geti2p.net.

[105] ithilgore. "Exploiting TCP and the Persist Timer Infiniteness." In: *Phrack Magazine* 0x0d (0x42) (June 2009).

[106] R. Jain, D. Chiu, and W. Hawe. *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems*. DEC Research Report TR-301. Maynard, MA, USA: Digital Equipment Corporation, Sept. 1984, p. 38.

[107] R. Jansen, K. Bauer, N. Hopper, and R. Dingledine. "Methodically Modeling the Tor Network." In: *CSET '12: Proceedings of the 5th Workshop on Cyber Security Experimentation and Test*. Bellevue, WA, USA, Aug. 2012.

[108] R. Jansen, J. Geddes, C. Wacek, M. Sherr, and P. F. Syverson. "Never Been KIST: Tor's Congestion Management Blossoms with Kernel-Informed Socket Transport." In: *USENIX Security '14: Proceedings of the 23rd USENIX Security Symposium*. San Diego, CA, USA, Aug. 2014, pp. 127–142.

[109] R. Jansen and N. Hopper. "Shadow: Running Tor in a Box for Accurate and Efficient Experimentation." In: *NDSS '12: Proceedings of the Network and Distributed System Security Symposium*. San Diego, CA, USA, Feb. 2012.

[110] R. Jansen, P. F. Syverson, and N. Hopper. "Throttling Tor Bandwidth Parasites." In: *USENIX Security '12: Proceedings of the 21th USENIX Security Symposium*. Bellvue, WA, USA, Aug. 2012, pp. 349–363.

[111] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. F. Syverson. "Users get routed: traffic correlation on tor by realistic adversaries." In: *CCS '13: Proceedings of the 20th ACM Conference on Computer and Communications Security*. Berlin, Germany, Oct. 2013, pp. 337–348.

[112] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. "An efficient rate allocation algorithm for ATM networks providing max-min fairness." In: *HPN '95: Proceedings of the Sixth International Conference on High Performance Networking*. Palma de Mallorca, Spain, Sept. 1995, pp. 143–154.

[113] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. "Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions." In: *SIGMETRICS '98 / PERFORMANCE '98: Joint International Conference on Measurement and Modeling of Computer Systems*. Madison, WI, USA, June 1998, pp. 78–89.

[114] M. Kenney. *Ping of Death*. `http://insecure.org/sploits/ping-o-death.html`.

[115] S. Khanvilkar and A. Khokhar. "Virtual Private Networks: An Overview with Performance Evaluation." In: *IEEE Communications Magazine* 42 (10) (Oct. 2004), pp. 146–154.

[116] C. Kiraly, G. Bianchi, and R. Lo Cigno. *Solving Performance Issues in Anonymization Overlays with a L3 Approach*. Tech. rep. DISI-08-041, Ver. 1.1. University of Trento, Italy, Sept. 2008.

[117] C. Kiraly and R. L. Cigno. "IPsec-Based Anonymous Networking: A Working Implementation." In: *ICC '09: Proceedings of the IEEE International Conference on Communications*. Dresden, Germany, June 2009, pp. 1–5.

[118] F. Klemm, J.-Y. L. Boudec, and K. Aberer. "Congestion Control for Distributed Hash Tables." In: *NCA '06: Proceedings of the 5th IEEE International Symposium on Network Computing and Applications*. Cambridge, MA, USA, July 2006, pp. 189–195.

[119] S. Köpsell. "Low Latency Anonymous Communication – How Long Are Users Willing to Wait?" In: *ETRICS '06: Proceedings of the International Conference on Emerging Trends in Information and Communication Security*. Freiburg, Germany, June 2006, pp. 221–237.

[120] H. T. Kung, T. Blackwell, and A. Chapman. "Credit-based flow control for ATM networks: credit update protocol, adaptive credit allocation and statistical multiplexing." In: *SIGCOMM '94: Proceedings of the 1994 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. London, UK, Aug. 1994.

[121] A. Kuzmanovic and E. W. Knightly. "Low-rate TCP-targeted Denial of Service Attacks and Counter Strategies." In: *IEEE/ACM TON* 14 (4) (2006).

[122] G.-I. Kwon and J. W. Byers. "ROMA: Reliable Overlay Multicast with Loosely Coupled TCP Connections." In: *INFOCOM '04: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*. Hong Kong, China, Mar. 2004, pp. 385–395.

[123] R. J. La, J. Walrand, and V. Anantharam. *Issues in TCP Vegas*. Tech. rep. M99/3. University of California at Berkeley, Jan. 1999. URL: www.eecs.berkeley.edu/~ananth/1999-2001/Richard/IssuesInTCPVegas.pdf.

[124] P. Lai, S. Yiu, K. Chow, C. Chong, and L. Hui. "An efficient Bloom Filter Based Solution for Multiparty Private Matching." In: *SAM '06: Proceedings of the 2006 International Conference on Security & Management*. Las Vegas, Nevada, USA, June 2006, pp. 286–292.

[125] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. *SOCKS Protocol Version 5*. RFC 1928 (Proposed Standard). Internet Engineering Task Force, Mar. 1996.

[126] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. B. Postel, L. G. Roberts, and S. S. Wolff. "A brief history of the internet." In: *Computer Communication Review* 39 (5) (2009), pp. 22–31.

[127] D. J. Leith, R. Shorten, and G. McCullagh. "Experimental evaluation of Cubic-TCP." In: *PFLDnet '08: Proceedings of the 6th International Workshop on Protocols for Fast Long-Distance Networks*. Manchester, U.K., Mar. 2008.

[128] *Libevent*. URL: http://monkey.org/~provos/libevent/.

[129] P. Lieven and B. Scheuermann. "High-Speed Per-Flow Traffic Measurement with Probabilistic Multiplicity Counting." In: *INFOCOM '10: Proceedings of the 29th Annual Joint Conference of the IEEE Computer and Communications Societies*. San Diego, CA, USA, Mar. 2010.

[130] C. Lochert, B. Scheuermann, and M. Mauve. "A Probabilistic Method for Cooperative Hierarchical Aggregation of Data in VANETs." In: *Elsevier Ad Hoc Networks* 8 (5) (July 2010), pp. 518–530.

[131] K. Loesing. *Implement new metric on bidirectional use of connections*. Aug. 2010. URL: https://trac.torproject.org/projects/tor/ticket/1819.

[132] K. Loesing. "Measuring the Tor Network from Public Directory Information." In: *HotPETs '09: 2nd Workshop on Hot Topics in Privacy Enhancing Technologies*. Seattle, Washington, USA, Aug. 2009.

[133] K. Loesing. *Remove data structure containing unique IP address sets*. Mar. 2015. URL: https://trac.torproject.org/projects/tor/ticket/15469.

[134] K. Loesing, S. J. Murdoch, and R. Dingledine. "A Case Study on Measuring Statistical Data in the Tor Anonymity Network." In: *WECSR '10: Workshop on Ethics in Computer Security Research*. Tenerife, Canary Islands, Spain, Jan. 2010.

[135] K. Loesing, S. J. Murdoch, and R. Jansen. *Evaluation of a libutp-based Tor datagram implementation*. Tech. rep. 2013-10-001. The Tor Project, Oct. 2013. URL: https://research.torproject.org/techreports/libutp-2013-10-30.pdf.

[136] H. V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. *iPlane: An Information Plane for Distributed Services*. http://iplane.cs.washington.edu.

[137] I. Maki, G. Hasegawa, M. Murata, and T. Murase. "Performance analysis and improvement of TCP proxy mechanism in TCP overlay networks." In: *ICC '05: Proceedings of the IEEE International Conference on Communications*. Seoul, Korea, May 2005, pp. 184–190.

[138] D. Many, M. Burkhart, and X. Dimitropoulos. *Fast Private Set Operations with SEPIA*. Tech. rep. 345. ETH Zurich, Mar. 2012. URL: http://www.sepia.ee.ethz.ch/publications/setops%5C_TIK-%20Report-345.pdf.

[139] V. P. Marco Valerio Barbera Vasileios P. Kemerlis and A. Keromytis. "CellFlood: Attacking Tor Onion Routers on the Cheap." In: *ESORICS '13: Proceedings of the 18th European Symposium on Research in Computer Security*. Egham, U.K., Sept. 2013.

[140] D. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Halpern. "Worst-case background knowledge for privacy-preserving data publishing." In: *ICDE '07: Proceedings of the 23rd IEEE International Conference on Data Engineering*. Istanbul, Turkey, Apr. 2007, pp. 126–135.

[141] N. Mathewson. *We should have better, fairer OOM handling*. June 2013. URL: https://trac.torproject.org/projects/tor/ticket/9093.

[142] S. Mauw, J. Verschuren, and E. P. de Vink. "A Formalization of Anonymity and Onion Routing." In: *ESORICS '04: Proceedings of the 9th European Symposium on Research in Computer Security*. Sophia Antipolis, France, Sept. 2004, pp. 109–124.

[143] D. McCoy, K. S. Bauer, D. Grunwald, T. Kohno, and D. C. Sicker. "Shining Light in Dark Places: Understanding the Tor Network." In: *PETS '08: Proceedings of the 8th Privacy Enhancing Technologies Symposium*. Leuven, Belgium, July 2008, pp. 63–76.

[144] G. Minshall. *Tcpdpriv*. 1996. URL: http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html.

[145] N. Mishra and M. Sandler. "Privacy via Pseudorandom Sketches." In: *PODS '06: Proceedings of the 25st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. Chicago, IL, USA, June 2006, pp. 143–152.

[146] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov. "Stealthy Traffic Analysis of Low-Latency Anonymous Communication Using Throughput Fingerprinting." In: *CCS '11: Proceedings of the 18th ACM Conference on Computer and Communications Security*. Chicago, IL, USA, Oct. 2011.

[147] J. C. Mogul. "Observing TCP Dynamics in Real Networks." In: *SIGCOMM '92: Proceedings of the 1992 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. Baltimore, Maryland, USA, Aug. 1992, pp. 305–317.

[148] S. Muddu, C. Tryfonas, F. M. Chiussi, and V. P. Kumar. "Max-Min Rate Control Algorithm for Available Bit Rate Service in ATM Networks." In: *ICC '96: Proceedings of the IEEE International Conference on Communications*. Dallas, TX, USA, June 1996, pp. 412–418.

[149] S. J. Murdoch. *Comparison of Tor Datagram Designs*. Tech. rep. 2011-11-001. The Tor Project, Nov. 2011. URL: https://research.torproject.org/techreports/datagram-comparison-2011-11-07.pdf.

[150] S. J. Murdoch and G. Danezis. "Low-Cost Traffic Analysis of Tor." In: *SP '05: Proceedings of the 26th IEEE Symposium on Security and Privacy*. Oakland, CA, USA, May 2005, pp. 183–195.

[151] S. J. Murdoch and P. Zielinski. "Sampled Traffic Analysis by Internet-Exchange-Level Adversaries." In: *PET '07: Proceedings of the 7th Workshop on Privacy Enhancing Technologies*. Ottawa, ON, Canada, June 2007, pp. 167–183.

[152] J. Nagle. *Congestion Control in IP/TCP Internetworks*. RFC 896. Internet Engineering Task Force, Jan. 1984.

[153] M. F. Nowlan, N. Tiwari, J. R. Iyengar, S. O. Amin, and B. Ford. "Fitting Square Pegs Through Round Pipes: Unordered Delivery Wire-Compatible with TCP and TLS." In: *NSDI '12: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation*. San Jose, CA, USA, Apr. 2012, pp. 383–398.

[154] M. F. Nowlan, D. I. Wolinsky, and B. Ford. "Reducing Latency in Tor Circuits with Unordered Delivery." In: *FOCI '13: Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*. Washington, DC, USA, Aug. 2013.

[155] *OOM Killer*. http://linux-mm.org/OOM_Killer.

[156] L. Øverlier and P. Syverson. "Locating Hidden Servers." In: *SP '06: Proceedings of the 27th IEEE Symposium on Security and Privacy*. Oakland, CA, USA, May 2006, pp. 100–114.

[157] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. "Modeling TCP Throughput: A Simple Model and its Empirical Validation." In: *SIGCOMM '98: Proceedings of the 1998 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. Vancouver, BC, Canada, Aug. 1998, pp. 303–314.

[158] O. Papapetrou, W. Siberski, and W. Nejdl. "Cardinality estimation and dynamic length adaptation for Bloom filters." In: *Distributed and Parallel Databases* 28 (2) (2010), pp. 119–156.

[159] V. Pappas, E. Athanasopoulos, S. Ioannidis, and E. P. Markatos. "Compromising Anonymity Using Packet Spinning." In: *ISC 08: Proceedings of the 11th Information Security Conference*. Taipei, Taiwan, Sept. 2008.

[160] A. Pathak, A. Wang, C. Huang, A. G. Greenberg, Y. C. Hu, R. Kern, J. Li, and K. W. Ross. "Measuring and Evaluating TCP Splitting for Cloud Services." In: *PAM '10: Proceedings of the 11th International Conference on Passive and Active Measurement*. Zurich, Switzerland, Apr. 2010, pp. 41–50.

[161] V. Paxson, M. Allman, J. Chu, and M. Sargent. *Computing TCP's Retransmission Timer*. RFC 6298 (Proposed Standard). Internet Engineering Task Force, June 2011.

[162] A. Pfitzmann and M. Hansen. *Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management – A Consolidated Proposal for Terminology*. Version v0.31. Feb. 2008. URL: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.31.pdf.

[163] H. Polat and W. Du. "Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques." In: *ICDM '03: Proceedings of the 3rd International Conference on Data Mining*. Melbourne, FL, USA, Dec. 2003, pp. 625–628.

[164] R. Pries, W. Yu, S. Graham, and X. Fu. "On performance bottleneck of anonymous communication networks." In: *IPDPS '08: IEEE International Parallel and Distributed Processing Symposium*. Miami, FL, USA, Apr. 2008.

[165] K. Ramakrishnan, S. Floyd, and D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168 (Proposed Standard). Internet Engineering Task Force, Sept. 2001.

[166] J. Reardon and I. Goldberg. "Improving Tor using a TCP-over-DTLS Tunnel." In: *USENIX Security '09: Proceedings of the 18th USENIX Security Symposium*. Montreal, Canada, Aug. 2009.

[167] M. K. Reiter and A. D. Rubin. "Crowds: Anonymity for Web Transactions." In: *ACM Transactions on Information and System Security* 1 (1) (June 1998), pp. 66–92.

[168] M. Rennhard and B. Plattner. "Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection." In: *WPES '02: Proceedings of the ACM Workshop on Privacy in the Electronic Society*. Washington, DC, USA, Nov. 2002, pp. 91–102.

[169] E. Rescorla. *Diffie-Hellman Key Agreement Method*. RFC 2631 (Proposed Standard). Internet Engineering Task Force, June 1999.

[170] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347 (Proposed Standard). Internet Engineering Task Force, Jan. 2012.

[171] T. Ries, R. State, and A. Panchenko. "Comparison of Low-Latency Anonymous Communication Systems - Practical Usage and Performance." In: *AISC '11: Proceedings of the 9th Australasian Information Security Conference*. Perth, Australia, Jan. 2011, pp. 77–86.

[172] M. Roughan and Y. Zhang. "Secure distributed data-mining and its application to large-scale network measurements." In: *SIGCOMM Comput. Commun. Rev.* 36 (1) (Jan. 2006), pp. 7–14.

[173] RSnake. *Slowloris HTTP DoS*. `http://ha.ckers.org/slowloris/`.

[174] *R-U-Dead-Yet (RUDY)*. `https://code.google.com/p/r-u-dead-yet/`.

[175] P. Samarati. "Protecting Respondents' Identities in Microdata Release." In: *IEEE Transactions on Knowledge and Data Engineering* 13 (6) (Nov. 2001), pp. 1010–1027.

[176] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. "TCP Congestion Control with a Misbehaving Receiver." In: *ACM SIGCOMM CCR* 29 (5) (1999).

[177] R. Schnell, T. Bachteler, and J. Reiher. "Privacy-preserving record linkage using Bloom filters." In: *BMC Medical Informatics and Decision Making* 9 (1) (2009), p. 41.

[178] A. Serjantov and G. Danezis. "Towards an Information Theoretic Metric for Anonymity." In: *PET '02: Proceedings of Privacy Enhancing Technologies Workshop*. San Francisco, CA, USA, Apr. 2002, pp. 259–263.

[179] A. Serjantov and P. Sewell. "Passive Attack Analysis for Connection-Based Anonymity Systems." In: *ESORICS '03: Proceedings of the 8th European Symposium on Research in Computer Security*. Gjøvik, Norway, Oct. 2003, pp. 116–131.

[180] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind. *Low Extra Delay Background Transport (LEDBAT)*. RFC 6817 (Experimental). Internet Engineering Task Force, Dec. 2012.

[181] S. Shalunov. *Netkill – generic remote DoS attack*. http://seclists.org/bugtraq/2000/Apr/152. 2000.

[182] A. Shamir. "How to share a secret." In: *Communications of the ACM* 22 (11) (Nov. 1979), pp. 612–613.

[183] R. Sherwood, B. Bhattacharjee, and R. Braud. "Misbehaving TCP Receivers Can Cause Internet-wide Congestion Collapse." In: *CCS '05: Proceedings of the 12th ACM Conference on Computer and Communications Security*. Alexandria, VA, USA, Nov. 2005.

[184] F. Shirazi, M. Goehring, and C. Díaz. "Tor Experimentation Tools." In: *SPW '15: Proceedings of the 36th IEEE Symposium on Security and Privacy Workshops*. San Jose, CA, USA, May 2015, pp. 206–213.

[185] M. Smart, G. R. Malan, and F. Jahanian. "Defeating TCP/IP Stack Fingerprinting." In: *USENIX Security '00: Proceedings of the 9th USENIX Security Symposium*. Denver, CO, USA, Aug. 2000.

[186] D. X. Song, D. Wagner, and X. Tian. "Timing Analysis of Keystrokes and Timing Attacks on SSH." In: *USENIX Security '01: Proceedings of the 10th USENIX Security Symposium*. Washington, DC, USA, Aug. 2001.

[187] L. Strigeus, G. Hazel, S. Shalunov, A. Norberg, and B. Cohen. *BEP 29: µTorrent transport protocol*. June 2009. URL: www.bittorrent.org/beps/bep%5C_0029.html.

[188] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. "Towards an Analysis of Onion Routing Security." In: *PET '00: Proceedings of the International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. Berkeley, CA, USA, July 2000.

[189] C. Tang and I. Goldberg. "An improved algorithm for Tor circuit scheduling." In: *CCS '10: Proceedings of the 17th ACM Conference on Computer and Communications Security*. Chicago, IL, USA, Oct. 2010, pp. 329–339.

[190] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias. "Spatio-Temporal Aggregation Using Sketches." In: *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*. Boston, MA, USA, Mar. 2004, pp. 214–226.

[191] *tcp – Linux man page*. URL: http://linux.die.net/man/7/tcp.

[192] W. W. Terpstra, C. Leng, M. Lehn, and A. Buchmann. "Channel-Based Unidirectional Stream Protocol (CUSP)." In: *INFOCOM '10 Mini Conference: Proceedings of the 29th Annual Joint Conference of the IEEE Computer and Communications Societies Mini Conference*. San Diego, CA, USA, Mar. 2010.

[193] The Tor Project. *TC: A Tor control protocol (Version 1)*. URL: https://gitweb.torproject.org/torspec.git/tree/control-spec.txt.

[194] The Tor Project. *Tor directory protocol, version 3*. URL: https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt.

[195] The Tor Project. *Tor Metrics Portal*. https://metrics.torproject.org.

[196] D. H. K. Tsang and W. K. F. Wong. "A New Rate-Based Switch Algorithm for ABR Traffic to Achieve Max-Min Fairness with Analytical Approximation Delay Adjustment." In: *INFOCOM '96: Proceedings of the 15th Annual Joint Conference of the IEEE Computer and Communications Societies*. an Francisco, CA, USA, Mar. 1996, pp. 1174–1181.

[197] G. Urvoy-Keller and E. W. Biersack. "A Congestion Control Model for Multicast Overlay Networks and its Performance." In: *NGC '02: Proceedings of the 4th International Workshop on Networked Group Communication*. Boston, MA, USA, Oct. 2002, pp. 141–147.

[198] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. "State-of-the-art in privacy preserving data mining." In: *ACM SIGMOD Record* 33 (1) (Mar. 2004), pp. 50–57.

[199] C. Viecco. "UDP-OR: A Fair Onion Transport Design." In: *HotPETS '08: 1st Workshop on Hot Topics in Privacy Enhancing Technologies*. Leuven, Belgium, July 2008.

[200] T. Wang, K. S. Bauer, C. Forero, and I. Goldberg. "Congestion-Aware Path Selection for Tor." In: *FC '12: Proceedings of the 16th International Conference on Financial Cryptography and Data Security*. Bonaire, Mar. 2012, pp. 98–113.

[201] R. Wendolsky, D. Herrmann, and H. Federrath. "Performance Comparison of low-latency Anonymisation Services from a User Perspective." In: *PET '07: Proceedings of the 7th Workshop on Privacy Enhancing Technologies*. Ottawa, ON, Canada, June 2007, pp. 233–253.

[202] K.-y. Whang, B. T. Vander-Zanden, and H. M. Taylor. "A Linear-Time Probabilistic Counting Algorithm for Database Applications." In: *ACM Transactions on Database Systems* 15 (2) (June 1990), pp. 208–229.

[203] P. Winter, R. Köwer, M. Mulazzani, M. Huber, S. Schrittwieser, S. Lindskog, and E. R. Weippl. "Spoiled Onions: Exposing Malicious Tor Exit Relays." In: *PETS '14: Proceedings of the 14th Workshop on Privacy Enhancing Technologies*. Amsterdam, Netherlands, July 2014, pp. 304–331.

[204] M. Wright, M. Adler, B. N. Levine, and C. Shields. "Defending anonymous communications against passive logging attacks." In: *SP '03: Proceedings of the 24th IEEE Symposium on Security and Privacy*. Oakland, CA, USA, May 2003, pp. 28–41.

[205] F. Xie, N. Jiang, Y. H. Ho, and K. A. Hua. "Semi-Split TCP: Maintaining End-to-End Semantics for Split TCP." In: *LCN '07: Proceedings of the 32nd Annual IEEE International Conference on Local Computer Networks*. Dublin, Ireland, Oct. 2007, pp. 303–314.

[206] C.-F. Yu and V. D. Gligor. "A Formal Specification and Verification Method for the Prevention of Denial of Service." In: *SP '88: Proceedings of the 9th IEEE Symposium on Security and Privacy*. Oakland, CA, USA, May 1988.

[207] L. Zhang and D. Clark. "Oscillating behavior of network traffic: A case study simulation." In: *Internetworking: research and experience* 1 (2) (1990), pp. 101–112.

[208] L. Zhang, S. Shenker, and D. D. Clark. "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic." In: *SIGCOMM '91: Proceedings of the 1991 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. Zurich, Switzerland, Sept. 1991, pp. 133–147.